Design Document for R Package glmm

Christina Knudson

April 15, 2018

Abstract

This design document describes the process of performing Monte Carlo maximum likelihood (MCML) for generalized linear mixed models. First, penalized quasi-likelihood (PQL) estimates are calculated, which help generate simulated random effects. Then, the Monte Carlo likelihood approximation (MCLA) is calculated using the simulated random effects. Next, the MCLA is maximized to find Monte Carlo maximum likelihood estimates, the corresponding Fisher Information, and other statistics. Additional inference is then possible, including confidence intervals for the parameters.

1 Theory

Let $y = (y_1, \ldots, y_n)'$ be a vector of observed data. Let $u = (u_1, \ldots, u_q)'$ be a 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. MCLA theoretically works for any $\tilde{f}(u)$, but the $\tilde{f}(u)$ chosen for this package is the mixture distribution specified in section 5.3.

Then the Monte Carlo log likelihood approximation is

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

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

When $\tilde{f}$ does not depend on θ, the gradient vector of the MCLA with respect to θ is

$$\nabla l_m(\theta) = \frac{\sum_{k=1}^{m} \nabla \log f_\theta(u_k, y) \frac{f_\theta(u_k, y)}{\tilde{f}(u_k)}}{\sum_{k=1}^{m} \frac{f_\theta(u_k, y)}{\tilde{f}(u_k)}}, \tag{4}$$

and the Hessian matrix of the MCLA is

$$\nabla^2 l_m(\theta) = \frac{\sum_{k=1}^{m} \nabla^2 \log f_\theta(y, u_k) \frac{f_\theta(y, u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^{m} \frac{f_\theta(y, u_k)}{\tilde{f}(u_k)}} + \frac{\sum_{k=1}^{m} \left[\nabla \log f_\theta(y, u_k) - \nabla l_m(\theta)\right] \left[\nabla \log f_\theta(y, u_k) - \nabla l_m(\theta)\right]' \frac{f_\theta(y, u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^{m} \frac{f_\theta(y, u_k)}{\tilde{f}(u_k)}} \tag{5}$$

Details for these derivatives can be found in section A.1. Calculation for the gradient and Hessian when $\tilde{f}$ is dependent on θ can be found in section A.2.

Now any likelihood-based inference, such as maximum likelihood, can be performed on $l_m(\theta)$ and its derivatives.

2 Model fitting function

The model fitting function is be the primary function the user uses. The user specifies the response and the predictors using the R formula mini-language as interpreted by `model.matrix`. Let n be the observed sample size and recall that p is the number of fixed effects. As a result of the user specifying the fixed effects, the model fitting function creates matrix X, which has dimensions $n \times p$.

The user specifies the random effects in the same way as the fixed effects. This is also how users of `reaster` (of R package `aster` (**?**)) specify the random effects. That is, random effects are expressed using the R formula mini-language. Let q_t be the number of random effects associated with variance component ν_t. When the random effects are specified, a list of T model matrices are created. The tth model matrix Z_t has dimensions $n \times q_t$.

The user also specifies the expontial family (details in section 3), the name of the data set, and the names of the variance components.

Thus, the following is a sample command with fixed predictors x_1 and x_2 and with random effects created by categorical variables *school* and *classroom*:

```
glmm(y ~ x1+ x2, list(~0+school,~0+classroom),  family.glmm="bernoulli.glmm",
data=schooldat,varcomps.names=c("school","classroom"),varcomps.equal=c(1,2),
debug=FALSE)
```

It is possible that the user could want some variance components to be set equal. For example, **?**'s (**?**) influenza dataset contains four years with random effects for each year. The authors want the within-year variance components to be equal. There are also variance components for subject-specific intercepts and for the decreased susceptibility to illness in year 4 (since the strain of flu during year 4 was a repeat of a previous year). Suppose `year` is a categorical variable with four levels, and `year1` through `year4` are dummy variables. Thus, the call contains these arguments:

```
glmm(y~year,list(~0+subject,~0+year1,~0+year2,~0+repeat,~0+year3,~0+year4),
 varcomps.equal=c(1,2,2,3,2,2), varcomps.names=c("subject","year","repeat"),
data=flu,family.glmm=bernoulli.glmm)
```

Initially, `model.matrix` makes Z into a list of 6 design matrices. Since we have 3 distinct variance components, we want 3 design matrices. We take the design matrices that share a variance component and use `cbind` to bind them together. The result is 3 model matrices in the Z list, one for each variance component. We want to put them in order (1,2,3 according to `varcomps.equal`) so that the names for each model matrix ("subject","year","repeat") are correctly assigned. The variance estimates are eventually listed in the model summary in this same order as well.

After interpreting the model the user has specified, the next step is to find penalized quasi-likelihood (PQL) estimates. The process of finding these estimates is detailed in section 8. The PQL estimates parameterize the importance sampling distribution $\tilde{f}(u_k)$ which generate the random effects. More information on generating the random effects is in section 5.3.

Next, `trust` is implemented to maximize the MCLA objective function (details on the objective function are in section 6). Finally, the function returns parameter estimates, the log likelihood

evaluated at those estimates, the gradient vector of the log likelihood, the Hessian of the log likelihood at those estimates, information from the `trust` optimization, and other information.

3 Families

Let g be the canonical link function and μ be a vector of length n such that

$$g(\mu) = X\beta + Zu \tag{6}$$

The choice of the link function is related to the distribution of the data, $f_\theta(y|u)$. If the data have a Bernoulli distribution, the link is $\text{logit}(\mu)$. If the data have a Poisson distribution, the link is $\log(\mu)$. Currently, `glmm` offers only these two choices for the family, but any exponential family could work and can be easily added later.

For simplicity of future notation, let $\eta = g(\mu) = X\beta + Zu$. Let $c(\eta)$ denote the cumulant function such that the log of the data density can be written as

$$y'\eta - c(\eta) = \sum_i [y_i\eta_i - c(\eta_i)] \tag{7}$$

The user is required to specify the family in the model-fitting function. Once the family is specified, many family-specific functions are called. They are contained in an S3 class called "glmm.family". Each family function outputs a list including the family name (a character string such as "bernoulli.glmm"), a function that calculates the value of the cumulant function $c(\eta)$, a function that calculates the cumulant's first derivative $c'(\eta)$ with the derivative taken with respect to η, and a function that calculates the cumulant's second derivative $c''(\eta)$.

The users provide the family in the model-fitting function by either enter the character string ("bernoulli.glmm"), the function (`bernoulli.glmm()`), or the result of invoking the function. The following code for using the input to determine the family is adapted from `glm`.

```
logDensity<-function(family.glmm)
{
if(is.character(family.glmm))
family.glmm<-get(family.glmm,mode="function",envir=parent.frame())
if(is.function(family.glmm))
family.glmm<-family.glmm()
if(!inherits(family.glmm,"glmm.family"))
stop(" 'family.glmm' not recognized")
return(family.glmm)
}
```

We interpret this as follows. If the user has entered the family as a string, go get the R object with that family name, either from the immediate environment or the parent environment. If this has happened, `family.glmm` is now a function. If `family.glmm` is a function (either because the user entered it as a function or because of the preceding step), invoke that function. At this point, `family.glmm` should have class "`glmm.family`." If this is not the case (maybe because of a typo or maybe because they entered "`poisson`" rather than "`poisson.glmm`"), then stop and return an error.

With this family of functions, calculating $c(\eta_i), c'(\eta_i)$, and $c''(\eta_i)$ is as simple as:

```
family.glmm$cum(args)
family.glmm$cp(args)
family.glmm$cpp(args)
```

For the Bernoulli distribution, we calculate these values (`cum`, `cp`, and `cpp`) as follows. In order to be careful with our computer arithmetic, we use the `log1p` function and the second set of equalities.

$$c(\eta_i) = \log(1 + e^{\eta_i}) = \begin{cases} \log(1 + e^{\eta_i}) & \text{if } \eta_i \le 0, \\ \eta_i + \log(e^{-\eta_i} + 1) & \text{if } \eta_i > 0, \end{cases} \tag{8}$$

$$c'(\eta_i) = \frac{e^{\eta_i}}{1 + e^{\eta_i}} = \frac{1}{1 + e^{-\eta_i}} \tag{9}$$

$$c''(\eta_i) = \frac{e^{\eta_i}}{1 + e^{\eta_i}} - \frac{e^{2\eta_i}}{(1 + e^{\eta_i})^2} = \frac{1}{1 + e^{-\eta_i}} \cdot \frac{1}{1 + e^{\eta_i}} \tag{10}$$

For Poisson, these values (`cum`, `cp`, and `cpp`) are

$$c(\eta_i) = e^{\eta_i} \tag{11}$$

$$c'(\eta_i) = e^{\eta_i} \tag{12}$$

$$c''(\eta_i) = e^{\eta_i}. \tag{13}$$

Then we use these pieces to create the scalar $c(\eta)$, the vector $c'(\eta)$ and the matrix $c''(\eta)$. We calculate

$$c(\eta) = \sum_i c(\eta_i). \tag{14}$$

The vector $c'(\eta)$ has components $c'(\eta_i)$. The matrix $c''(\eta)$ is diagonal with diagonal elements $c''(\eta_i)$.

In the R function `glm`, the user can choose the link. The canonical link must be used in the `glmm` package so that we have an exponential family. The canonical link is included in `family.glmm` in case the user does not know it already.

Also, this family of functions contains a check to ensure the data are valid given the family type. If `family.glmm` is `bernoulli.glmm`, the data should contain only 0 and 1. If `family.glmm` is `poisson.glmm`, then the data should be nonnegative. If the data set does not pass the check, the check returns an error message.

3.1 Redone in C

Redone in C, I have separate functions for calculating $c(\eta)$, $c'(\eta)$, $c''(\eta)$: `cum3.c`, `cp3.c`, and `cpp3.c`. The inputs for each function are identical: `eta` (an array of doubles), `neta` (the length of `eta`), the type (to denote the family: 1 indicates Bernoulli and 2 indicates Poisson), and an array of doubles to contain the result. These are passed in as pointers. The functions calculate the cumulant function or one of its first two derivatives. Each function contains a switch statement for the glmm family. The calculations for each of these functions have been shown earlier in this section. This function is type void: rather than returning the cumulant or its derivatives, the pointers are changed to contain the results. This function is invoked by `el`, described in section 4.

4 Log density of the data (`el`)

This section provides details for the log density of the data and two of its derivatives.

4.1 Equations

Recall the log of the data density is

$$\log f_\theta(y|u) = y'\eta - c(\eta) \tag{15}$$

$$= \sum_i y_i\eta_i - c(\eta_i) \tag{16}$$

where

$$\eta = X\beta + Zu. \tag{17}$$

The derivative of this with respect to one component, η_j, is

$$\frac{\partial}{\partial \eta_j} \log f_\theta(y|u) = y_j - c'(\eta_j). \tag{18}$$

The derivative of the component η_j with respect to one of the fixed effect predictors, β_l, is

$$\frac{\partial \eta_j}{\partial \beta_l} = X_{jl} \tag{19}$$

We'd like the derivative of the log of the data density with respect to β. The first step is using the chain rule as follows:

$$\frac{\partial}{\partial \beta_l} \log f_\theta(y|u) = \frac{\partial \eta_j}{\partial \beta_l} \frac{\partial}{\partial \eta_j} \log f_\theta(y|u) \tag{20}$$

$$= [y_j - c'(\eta_j)] X_{jl} \tag{21}$$

The mixed partial derivative (with respect to β_{l_1} and β_{l_2}) of the log data density can be written similarly:

$$\frac{\partial^2}{\partial \beta_{l_1} \partial \beta_{l_2}} \log f_\theta(y|u) = \frac{\partial}{\partial \beta_{l_2}} \left([y_j - c'(\eta_j)] X_{jl_1} \right) \tag{22}$$

$$= \frac{\partial \eta_j}{\partial \beta_{l_2}} \frac{\partial}{\partial \eta_j} \left([y_j - c'(\eta_j)] X_{jl_1} \right) \tag{23}$$

$$= -X_{jl_1} X_{jl_2} c''(\eta_j) \tag{24}$$

Letting $c'(\eta)$ be a vector with components $c'(\eta_j)$, the first derivative of the log data density can be written in matrix form as:

$$\frac{\partial}{\partial \beta} \log f_\theta(y|u) = X' \left[y - c'(\eta) \right]. \tag{25}$$

Letting $c''(\eta)$ be a diagonal matrix with components $c''(\eta_j)$, the second derivative of the log data density can be written in matrix form as:

$$\frac{\partial^2}{\partial \beta^2} \log f_\theta(y|u) = X'[-c''(\eta)]X \tag{26}$$

4.2 Redone in C

The C function to calculate the value of the log data density and its two derivatives are called by reference. The following pointers are passed in: double `Y`, double `X`, int `nrowX`, int `ncolX`, double `eta`, int `family`, double `elval`, double `elgradient`, double `elhessian`. The pointers `elval`, `elgradient` and `elhessian` are zeros before `el.C` is invoked. Invoking `el.C` then places the calculatedvalue of the log data density and two derivatives into `elval`, `elgradient` and `elhessian`.

The function `el.C` calls the following C functions: `cum3.C` to calculate the cumulant given a value of η, `cp3.C` to calculate the derivative of the cumulant given a value of η, `cpp3.C` to calculate the Hessian of the cumulant given a value of η, and functions to perform matrix multiplication.

5 Random effect generation and calculations

This section is focused on a vector of random effects u with length q. Section 5.1 details the relationship between ν and D. Section 5.2 states the importance sampling distribution. Section 5.3 describes the process of generating the random effects. Section 5.4 explains how to evaluate the distribution of the random effects and its first two derivatives.

5.1 Constructing D (`getEk`)

Recall that, for this version of the package, we assume that D is diagonal. Let ν be length T with components ν_t, and let $E_t, t = 1, \ldots, T$ be diagonal matrices with indicators on the diagonal so that $\sum_{t=1}^{T} E_t = I$. That is, the diagonal entries of E_t indicate whether that random effect has ν_t as its variance component. Then $D = \sum_{t=1}^{T} \nu_t E_t$.

Recall q_t is the number of random effects associated with variance component ν_t. Then q_t is also the number of nonzero entries in E_t and $q = \sum_{t=1}^{T} q_t$ is the total number of random effects in the model. The model fitting function of `glmm` has made `z` into a list with T design matrices (one for each distinct variance component). In order to create E_t, we need to count the number of columns (q_t) for each design matrix in the list. Since D is $q \times q$, each E_t is $q \times q$. E_1 has q_1 ones on the diagonal, followed by zeros to fill the rest of the diagonal. E_2 has q_1 zeros, then q_2 ones, then zeros for the rest of the diagonal. We continue this process to complete the construction of $E_t, t = 1, \ldots, T$.

5.2 Importance sampling distribution $\tilde{f}(u_k)$

Let s be a vector of length q that represents the random effects on the standard normal scale. That is s is defined such that, $u = D^{1/2}s$. Let σ be a vector of length T with components $\sqrt{\nu_t}, t = 1, \ldots, T$. Prior to generating random effects, the model fitting function of `glmm` has performed PQL and recorded the PQL estimates for β, σ and s. The PQL estimates are denoted by β^*, s^* and σ^*. Let

$$A^* = \sum_{t=1}^{T} E_t \sigma_t^* \tag{27}$$

and

$$D^* = A^* A^* \tag{28}$$

be matrices based on PQL estimates. We can "unstandardize" our PQL-based random effects:

$$u^* = A^* s^*. \tag{29}$$

Let p_1, p_2, p_3 be the proportions of the mixture distribution such that $p_1 + p_2 + p_3 = 1$. Let $f(u|\mu, \Sigma)$ denote the pdf for $u \sim N(\mu, \Sigma)$. Let $\grave{f}(u|0, D^*)$ denote the pdf for $u \sim t_\zeta$, a q-dimensional multivariate t with mean 0, scale matrix D^*, and ζ degrees of freedom. Then the importance sampling distribution is:

$$\tilde{f}(u) = p_1 \grave{f}(u|0, D^*) + p_2 f(u \,|\, u^*, D^*) + p_3 f(u \,|\, u^*, (Z'c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1}). \tag{30}$$

The first component of the mixture distribution t_ζ is chosen to ensure the gradient of the MCLA has a central limit theorem, which is proven in section B. The second component is chosen because it is centered at the PQL best guess of the random effect values u^* and has the PQL guess of the variance. The last component is centered at the PQL guess u^* and has a variance based on the Hessian of the PQL penalized likelihood. The idea of this last distribution is to generate random effects from a distribution whose Hessian matches that of the target distribution $f_\theta(y, u)$.

5.3 Generating random effects (genRand)

Recall that m is the overall Monte Carlo sample size. Let p_1, p_2, p_3 be the proportions of the mixture distribution such that $p_1 + p_2 + p_3 = 1$. We randomly determine m_1, m_2, m_3 according to the proportions p_1, p_2, p_3. More specifically, we generate m random numbers from a distribution that is uniform between 0 and 1. Then m_1 is the frequency with which the randomly generated numbers are between 0 and p_1, m_2 is the frequency with which the randomly generated numbers are between p_1 and $p_1 + p_2$, and $m_3 = m - p_1 - p_2$.

Once m_1, m_2 and m_3 are determined, we draw

$$u_k \sim t_\zeta, k = 1, \ldots, m_1 \tag{31}$$

$$u_k \sim N(u \,|\, u^*, D^*), k = m_1 + 1, \ldots, m_1 + m_2 \tag{32}$$

$$u_k \sim N(u \,|\, u^*, (Z'c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1}), k = m_1 + m_2 + 1, \ldots, m. \tag{33}$$

Details for drawing from a nonstandard normal are in section 5.3.2. We use R package `mvtnorm` to draw from a t distribution.

5.3.1 Finding the square root of a variance matrix

In order to generate draws from a nonstandard normal distribution, we need to calculate the square root of a variance matrix. When the variance matrix is diagonal (for example, D^*), we simply take the root of the diagonal elements. When the variance matrix is not diagonal (for example, in the second component of the mixture distribution), we can use eigendecomposition. Eigendecompositions take a little bit more time than Cholesky decompositions but are more stable.

Let Σ denote the nondiagonal variance matrix. Eigendecomposition of Σ provides orthogonal matrix O (containing the eigenvectors) and diagonal matrix Λ (with diagonal entries λ being the eigenvalues of Σ). Then

$$\Sigma^{1/2} = O\Lambda^{1/2}O'. \tag{34}$$

Finding $\Lambda^{1/2}$ is as easy as taking the root of the diagonal entries. We know that $\Sigma^{1/2}$ is correct because

$$\begin{aligned}\Sigma^{1/2}\Sigma^{1/2} &= O\Lambda^{1/2}O'O\Lambda^{1/2}O' \\ &= O\Lambda^{1/2}\Lambda^{1/2}O' \\ &= O\Lambda O' \\ &= \Sigma.\end{aligned} \tag{35}$$

5.3.2 Generating from nonstandard normal distributions

Construct $A^* = (D^*)^{1/2}$ and $((Z'c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1})^{1/2}$ as described in section 5.3.1.

Let $\breve{u}_k, k = m_1 + 1 \ldots, m$ be vectors of random effects that are drawn from the standard normal distribution. Then we center and scale $\breve{u}_k$ as follows:

$$u_k = u^* + A^*\,\breve{u}_k, k = m_1 + 1, \ldots, m_1 + m_2 \tag{36}$$

$$u_k = u^* + ((Z'c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1})^{1/2}\,\breve{u}_k, k = m_1 + m_2 + 1, \ldots, m. \tag{37}$$

The result or centering and scaling is that

$$u_k \sim N(u^*, D^*),\ k = m_1 + 1, \ldots, m_1 + m_2 \tag{38}$$

$$u_k \sim N(u^*, (Z'c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1}),\ k = m_1 + m2 + 1, \ldots, m. \tag{39}$$

5.4 Evaluating the distribution of random effects (`distRand`)

This section discusses evaluating the distribution of the random effects, the gradient of the random effects, and the Hessian of the random effects. That is, the equations in this section provide $\log f_\theta(u)$, $\nabla \log f_\theta(u)$, and $\nabla^2 \log f_\theta(u)$ for equation 62.

We can express the log unnormalized pdf of $N(0, D)$ as:

$$\log f_\theta(u) = \log f(u|0, D) = -q/2 \log(2\pi) - 1/2 \log|D| - (1/2)u'D^{-1}u \tag{40}$$

To find derivatives of this, we are going to use the diagonal form of D. Recall that for every ν_t, we can construct a matrix E_t (with dimensions the same as matrix D) that has 1s on the diagonal elements corresponding to the elements of D that contain ν_t and 0s elsewhere.

We can partition the random effects according to their variance components: $u = (U_1', ..., U_T')'$. Let D_t be the variance matrix for U_t. D_t has q_t rows and q_t columns. Thus D can be expressed as:

$$D = \begin{bmatrix} D_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_T \end{bmatrix} \tag{41}$$

Since D is diagonal, it follows that D^{-1} is also diagonal with diagonal entries $\frac{1}{\nu_1}$,...,$\frac{1}{\nu_T}$. Also, the assumption that D is diagonal makes calculating the determinant of D easy:

$$|D| = \nu_1^{q_1} ... \nu_T^{q_T} \tag{42}$$

Taking these two pieces of information into account allows us to write the log density for U as follows:

$$\log f_\theta(u) = -\frac{1}{2} \log |D| - \frac{1}{2} u' D^{-1} u \tag{43}$$

$$= -\frac{1}{2} \left[\sum_{t=1}^{T} q_t \log \nu_t \right] - \frac{1}{2} \sum_{t=1}^{T} \left[\frac{1}{\nu_t} U_t' U_t \right] \tag{44}$$

The first and second derivatives of each summand with respect to its associated ν_t are:

$$\frac{\partial}{\partial \nu_t} \log f_\theta(u_t) = -\frac{q_t}{2\nu_t} + \frac{1}{2\nu_t^2} U_t' U_t \tag{45}$$

and

$$\frac{\partial^2}{\partial \nu_t^2} \log f_\theta(u_t) = \frac{q_t}{2\nu_t^2} - \frac{1}{\nu_t^3} U_t' U_t. \tag{46}$$

Any other derivative is equal to 0. That is, for all $t_1 \neq t_2$,

$$\frac{\partial}{\partial \nu_{t_1}} \log f_\theta(u_{t_2}) = 0. \tag{47}$$

Also,

$$\frac{\partial}{\partial \beta} \log f_\theta(u_{t_2}) = 0. \tag{48}$$

Thus, if $\nu = (\nu_1, ..., \nu_T)$, the gradient of the random effects distribution is the following vector of length T:

$$\frac{\partial}{\partial \nu} \log f_\theta(u) = \left[-\frac{q_1}{2\nu_1} + \frac{1}{2\nu_1^2} U_1' U_1 \quad ... \quad -\frac{q_T}{2\nu_T} + \frac{1}{2\nu_T^2} U_T' U_T \right] \tag{49}$$

The Hessian matrix is the following diagonal matrix:

$$\frac{\partial^2}{\partial \nu^2} \log f_\theta(u) = \begin{bmatrix} \frac{q_1}{2\nu_1^2} - \frac{1}{\nu_1^3} U_1' U_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{q_T}{2\nu_T^2} - \frac{1}{\nu_T^3 U_T' U_T} \end{bmatrix} \tag{50}$$

To calculate the value, gradient, and Hessian, we need to provide ν, u, and the list `z` (from `mod.mcml`). The list `z` has T matrices, each with the number of columns equal to q_t. We need $q_t, t = 1, \ldots, T$ to calculate the log density and its derivatives.

The last thing we need to discuss is how to split U into $U_1, ..., U_T$. We know that the first q_1 items of U are U_1, the next q_2 items are U_2, etc. In other words, entries 1 through q_1 are U_1. Items $q_1 + 1$ through $q_1 + q_2$ are U_2, etc. We find these numbers (q_1, q_2, etc) from the number of columns of the items in the list `z`. This information of how to split u up is contained in vector `meow` and used by function `distRand3`. This is faster than recalculating `meow` for every call of `distRand3`.

Rewritten in C, this function takes the following as pointers: the double array `nu` that contains the variance components, the int `T` to specify the length of `nu`, the double array `mu` that contains the means of the random effects, the int array `nrandom` of length `T` that contains the number random effects from that variance component, the double array `Uvec` that contains one vector of generated random effects, the int array `meow` that specifies how to split `u` up based on the variance components, the double array `drgradient` that contains the resulting gradient, and the double array `drhessian` that contains the resulting Hessian.

The result of invoking the C function `distRand3` is the calculation of the gradient and Hessian for the distribution of the random effects. The value is evaluated by the C function `distRandGeneral`, described in section 5.5.1.

5.5 Evaluating the importance sampling distribution

This section discusses evaluating the importance sampling distribution shown in 30. The $\tilde{f}(u)$ calculation is used in equation 62. Section 5.5.1 describes how to evaluate the pdf for the normal components of $\tilde{f}(u)$. Section 5.5.2 describes how to evaluate the pdf for the t component of $\tilde{f}(u)$.

To be computationally stable, we need to be careful with the addition of the terms in 30 to prevent overflow. We know that

$$\tilde{f}(u) = p_1 \hat{f}(u|0, D^*) + p_2 f(u \,|\, u^*, D^*) + p_3 f(u \,|\, u^*, (Z' c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1}) \tag{51}$$

$$\begin{aligned} &= p_1 \exp(\log \hat{f}(u|0, D^*))) + p_2 \exp(\log(f(u \,|\, u^*, D^*)))+ \\ &+ p_3 \exp(\log(f(u \,|\, u^*, (Z' c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1}))). \end{aligned} \tag{52}$$

Set

$$\tilde{a} = \max\left\{\log \grave{f}(u|0, D^*)), \log f(u \mid u^*, D^*)), \log f(u \mid u^*, (Z'c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1})\right\} \quad (53)$$

Then

$$\begin{aligned}\tilde{f}(u) = p_1 \exp\left(\log \grave{f}(u|0, D^*) - \tilde{a}\right) + p_2 \exp\left(\log f(u \mid u^*, D^*) - \tilde{a}\right) + \\ + p_3 \exp\left[\log f\left(u \mid u^*, (Z'c''(X\beta^* + Zu^*)Z + (D^*)^{-1})^{-1}\right) - \tilde{a}\right]. \quad (54)\end{aligned}$$

is more stable computationally.

5.5.1 Normal distribution for general variance matrix Σ (`distRandGeneral`)

Let Σ be a variance matrix and let μ be a mean vector. Consider $N(\mu, \Sigma)$. We can write the log pdf of this distribution as:

$$\log f(u|\mu, \Sigma) = -\frac{q}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma^{-1}| - \frac{1}{2}(u-\mu)'\Sigma^{-1}(u-\mu) \quad (55)$$

The only part of this to discuss is $|\Sigma^{-1}|$. We can use eigendecomposition to make $\Sigma^{-1} = O\Lambda O'$ where O is orthogonal and Λ is the diagonal matrix with eigenvalues. Since orthogonal matrices have determinant ± 1, then $|O||O'| = 1$. Thus

$$\begin{aligned}|\Sigma^{-1}| &= |O'|\,|\Lambda|\,|O| &(56)\\ &= |O'|\,|O|\,|\Lambda| &(57)\\ &= |\Lambda|, &(58)\end{aligned}$$

which is just the product of the eigenvalues. The log of the determinant is calculated beforehand to save time, since this function is called $3m$ times throughout each trust optimization iteration, where m is again the Monte Carlo sample size.

This function is also rewritten in C with the following passed in as pointers: double `Sigma.inv` Σ^{-1}, double `logdet` $\log|\Sigma^{-1}|$, int `nrow`, double `uvec` a vector of random effects, double mu μ, and double `distRandGenVal`.

5.5.2 t distribution

Consider t_ζ. We can write the log unnormalized pdf of this distribution as:

$$\begin{aligned}\log \grave{f}(u|0, D^*) = \log\left(\Gamma\left(\frac{\zeta+q}{2}\right)\right) - \log\left(\Gamma\left(\frac{\zeta}{2}\right)\right) - \frac{q}{2}\log(\zeta\pi) &(59)\\ - \frac{1}{2}\log|D^*| - \left[\frac{\zeta+q}{2}\right]\log\left(1 + \frac{u'D^{*-1}u}{\zeta}\right) &(60)\\ &(61)\end{aligned}$$

The first four terms are constant with respect to u and can be calculated ahead of time. Rewritten in C, we need to pass in `tconstants` (the sum of the first four terms) as a double, `u` as an array of doubles, `zeta` as an int, `myq` as an int, the diagonal of `Dstarinv` as an array of doubles, and `logft` as a double. The final calculation will be written into `logft`.

I have another function (written in R) that calculates the constants for the t distribution. The arguments of this function are `zeta`, the diagonal of D^* `Dstardiag`, and the length of u `myq`.

6 Objective function: approximated log likelihood

The objective function is optimized by `trust` within the the model fitting function. In order to evaluate and maximize the approximated log likelihood, `trust` needs an objective function that returns the value, the gradient vector, and the Hessian matrix of the approximated log likelihood. Recall equations 3, 4, and 5 for calculating these quantities.

Denote

$$b_k = \log f_\theta(u_k) + \log f_\theta(y|u_k) - \log \tilde{f}(u_k). \tag{62}$$

For computational stability, we'll set $a = \max(b_k)$. Then the value of the MCLA is expressed as:

$$l_m(\theta) = a + \log\left[\frac{1}{m}\sum_{k=1}^{m} e^{b_k - a}\right] \tag{63}$$

Define the weights as:

$$v_\theta(u_k, y) = \frac{e^{b_k - a}}{\sum_{k=1}^{m} e^{b_k - a}} \tag{64}$$

Rewriting equation 4 using the notation for the weights, the gradient vector is:

$$\nabla l_m(\theta) = \sum_{k=1}^{m} \nabla \log f_\theta(u_k, y) v_\theta(u_k, y) \tag{65}$$

$$= \sum_{k=1}^{m} \nabla \left[\log f_\theta(y|u_k) + \log f_\theta(u_k)\right] v_\theta(u_k, y) \tag{66}$$

$$= \sum_{k=1}^{m} \left[\nabla \log f_\theta(y|u_k) + \nabla \log f_\theta(u_k)\right] v_\theta(u_k, y) \tag{67}$$

$$= \sum_{k=1}^{m} \left[\frac{\partial}{\partial \beta} \log f_\theta(y|u_k) \quad \frac{\partial}{\partial \nu} \log f_\theta(y|u_k)\right] v_\theta(u_k, y) + \left[\frac{\partial}{\partial \beta} \log f_\theta(u_k) \quad \frac{\partial}{\partial \nu} \log f_\theta(u_k)\right] v_\theta(u_k, y) \tag{68}$$

$$= \sum_{k=1}^{m} \left(\left[\frac{\partial}{\partial \beta} \log f_\theta(y|u_k) \quad 0\right] + \left[0 \quad \frac{\partial}{\partial \nu} \log f_\theta(u_k)\right] \right) v_\theta(u_k, y) \tag{69}$$

$$= \sum_{k=1}^{m} \left[\frac{\partial}{\partial \beta} \log f_\theta(y|u_k) \quad \frac{\partial}{\partial \nu} \log f_\theta(u_k)\right] v_\theta(u_k, y) \tag{70}$$

Then the Hessian matrix from equation 5 is expressed as:

$$\nabla^2 l_m(\theta) = \sum_{k=1}^{m} \nabla^2 \log f_\theta(y, u_k) v_\theta(u_k, y) \tag{71}$$

$$+ \sum_{k=1}^{m} \left[\nabla \log f_\theta(y, u_k) - \nabla l_m(\theta)\right] \left[\nabla \log f_\theta(y, u_k) - \nabla l_m(\theta)\right]' v_\theta(u_k, y).$$

Everything in this has already been defined except:

$$\nabla^2 \log f_\theta(u_k, y) = \begin{bmatrix} \frac{\partial^2}{\partial \beta^2} \log f_\theta(u_k, y) & \frac{\partial^2}{\partial \beta\, \partial \nu} \log f_\theta(u_k, y) \\ \frac{\partial^2}{\partial \beta\, \partial \nu} \log f_\theta(u_k, y) & \frac{\partial^2}{\partial \nu^2} \log f_\theta(u_k, y) \end{bmatrix} \tag{72}$$

$$= \begin{bmatrix} \frac{\partial^2}{\partial \beta^2} \log f_\theta(u_k) & \frac{\partial^2}{\partial \beta\, \partial \nu} \log f_\theta(u_k) \\ \frac{\partial^2}{\partial \beta\, \partial \nu} \log f_\theta(u_k) & \frac{\partial^2}{\partial \nu^2} \log f_\theta(u_k) \end{bmatrix} + \begin{bmatrix} \frac{\partial^2}{\partial \beta^2} \log f_\theta(y|u_k) & \frac{\partial^2}{\partial \beta\, \partial \nu} \log f_\theta(y|u_k) \\ \frac{\partial^2}{\partial \beta\, \partial \nu} \log f_\theta(y|u_k) & \frac{\partial^2}{\partial \nu^2} \log f_\theta(y|u_k) \end{bmatrix} \tag{73}$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & \frac{\partial^2}{\partial \nu^2} \log f_\theta(u_k) \end{bmatrix} + \begin{bmatrix} \frac{\partial^2}{\partial \beta^2} \log f_\theta(y|u_k) & 0 \\ 0 & 0 \end{bmatrix} \tag{74}$$

$$= \begin{bmatrix} \frac{\partial^2}{\partial \beta^2} \log f_\theta(y|u_k) & 0 \\ 0 & \frac{\partial^2}{\partial \nu^2} \log f_\theta(u_k) \end{bmatrix} \tag{75}$$

7 Summary of model

Typing `summary(mod)` provide a summary of the model. This is broken into two pieces (as is usually done for summaries in R): `summary.glmm` and `print.summary.glmm`. The first contains more calculated results than the second. When a user types `summary(mod)`, only the basic information is automatically printed. More information can be found in the summary list.

The summary performs all calculations and its value is a list of the following:

- the call.
- a matrix with the predictor name in the first column, the estimated coefficient in the second column, the standard error in the third column, the z value in the fourth column, and the two-sided p-value in the fifth column. All inference is asymptotic, so we use the standard normal distribution to calculate the p-values.
- a matrix with the variance estimate name in the first column, the estimated variance component in the second column, the standard error in the third column, the z value in the fourth column, and the one-sided p-value in the fifth column. All inference is asymptotic, so we use the standard normal distribution to calculate the p-values.
- the evaluated Monte Carlo log likelihood along with its first and second derivative.
- m_1, m_2, m_3.
- output from textttttrust, such as whether it converged. (See `trust` for more details.)
- the generated random effects.

A note on the standard errors: to calculate the standard errors, we take the MCLA Hessian matrix, invert it, take the diagonal elements, and apply the square root to them. It is possible that the Hessian will be noninvertible if it is close enough to singular that the computer thinks it is singular. Then the standard errors will all be infinite the user is warned.

Then `print.summary.glmm` prints the following:

- the call.
- a matrix with the predictor name in the first column, the estimated coefficient in the second column, the standard error in the third column, the z value in the fourth column, and the two-sided asymptotic p-value in the fifth column.
- a matrix with the variance estimate name in the first column, the estimated variance component in the second column, the standard error in the third column, the z value in the fourth

column, and the one-sided p-value in the fifth column. All inference is asymptotic, so we use the standard normal distribution to calculate the p-values.

Note that the summary and print.summary functions are S3 generic functions. This means is the user types `summary(mod)` and `summary` is a generic function. R checks the class of the model `mod` and automatically uses the `summary` and `print.summary` functions for that class of objects.

8 PQL

This section on PQL requires a change in notation so that we can avoid constrained optimization. Recall that $D = \mathrm{Var}(u)$ 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. The inner optimization is a well behaved quadratic while the outer optimization is more challenging. 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.

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. After defining

$$\eta = X\beta + ZAs \tag{76}$$

we calculate the log likelihood as

$$l(\eta) = y'\eta - c(\eta) \tag{77}$$

and the penalized likelihood as:

$$l(\eta) - \frac{1}{2}s's. \tag{78}$$

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

$$l(\eta) = \sum_i Y_i\eta_i - c(\eta_i) \tag{79}$$

then

$$\frac{\partial l(\eta)}{\partial \eta_i} = Y_i - c'(\eta_i) = Y_i - \mu_i. \tag{80}$$

Now we can write the following expression:

$$\begin{aligned}
\frac{\partial}{\partial \beta_k}\left[l(\eta) - \frac{1}{2}s's\right] &= \frac{\partial l(\eta)}{\partial \beta_k} && (81)\\
&= \frac{\partial l(\eta)}{\partial \eta_i}\frac{\partial \eta_i}{\partial \beta_k} && (82)\\
&= \sum_i (y_i - \mu_i)\frac{\partial \eta_i}{\partial \beta_k} && (83)\\
&= \sum_i (y_i - \mu_i)X_{ik} && (84)
\end{aligned}$$

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

$$\begin{aligned}
\frac{\partial}{\partial s}\left[l(\eta) - \frac{1}{2}s's\right] &= \frac{\partial l(\eta)}{\partial s} - \frac{1}{2}\frac{\partial s's}{\partial s} && (85)\\
&= \frac{\partial l(\eta)}{\partial \eta}\frac{\partial \eta}{\partial s} - s && (86)\\
&= (y-\mu)'\left[\frac{\partial}{\partial s}ZAs\right] - s && (87)\\
&= (y-\mu)'ZA - s && (88)
\end{aligned}$$

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

$$\frac{\partial}{\partial \beta}\left[l(\eta) - (1/2)s's\right] = X'(y-\mu) \tag{89}$$

$$\frac{\partial}{\partial s}\left[l(\eta) - (1/2)s's\right] = AZ'(y-\mu) - s \tag{90}$$

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.

$$\frac{\partial^2}{\partial s^2}\left[l(\eta) - (1/2)s's\right] = \frac{\partial}{\partial s}\left[(y - c'(\eta))'ZA - s\right] \tag{91}$$

$$= \left[-\frac{\partial}{\partial s}c'(\eta)\right]' ZA - I_q \tag{92}$$

$$= \left[-\frac{\partial c'(\eta)}{\partial \eta}\frac{\partial \eta}{\partial s}\right]' ZA - I_q \tag{93}$$

$$= \left[-c''(\eta)ZA\right]' ZA - I_q \tag{94}$$

$$= -AZ'\left[c''(\eta)\right] ZA - I_q \tag{95}$$

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.

$$\frac{\partial^2}{\partial \beta^2}\left[l(\eta) - (1/2)s's\right] = \frac{\partial}{\partial \beta}\left[X'(y - c'(\eta))\right] \tag{96}$$

$$= \frac{\partial}{\partial \beta}\left[X'y - X'(c'(\eta))\right] \tag{97}$$

$$= \frac{\partial}{\partial \beta}\left[-X'(c'(\eta))\right] \tag{98}$$

$$= -X\left[{}'\frac{\partial}{\partial \beta}c'(\eta)\right] \tag{99}$$

$$= -X'\left[\frac{\partial c'(\eta)}{\partial \eta}\frac{\partial \eta}{\partial \beta}\right] \tag{100}$$

$$= -X'\left[c''(\eta)\right] X \tag{101}$$

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

$$\frac{\partial^2}{\partial\beta\ \partial s}\left[l(\eta) - (1/2)s's\right] = \frac{\partial}{\partial\beta}\left\{\left[y - c'(\eta)\right]' ZA\right\} \tag{102}$$

$$= \frac{\partial}{\partial\beta}\left\{y'ZA - \left[c'(\eta)\right]' ZA\right\} \tag{103}$$

$$= -\frac{\partial}{\partial\beta}\left\{\left[c'(\eta)\right]' ZA\right\} \tag{104}$$

$$= -\left[\frac{\partial c'(\eta)}{\partial\beta}\right]' ZA \tag{105}$$

$$= -\left[\frac{\partial c'(\eta)}{\partial\eta}\frac{\partial\eta}{\partial\beta}\right]' ZA \tag{106}$$

$$= -\left[c''(\eta)X\right]' ZA \tag{107}$$

$$= -X'\left[c''(\eta)\right] ZA \tag{108}$$

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

$$\tilde{\eta} = X\tilde{\beta} + ZA\tilde{s} \tag{109}$$

$$l(\tilde{\eta}) = y'\tilde{\eta} - c(\tilde{\eta}) \tag{110}$$

$$A = \sum_k E_k\sigma_k. \tag{111}$$

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:

$$l(\tilde{\eta}) - \frac{1}{2}\tilde{s}'\tilde{s} - \frac{1}{2}\log|AZ'WZA + I| \tag{112}$$

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

$$\frac{1}{2}\log|AZ'WZA + I| = \frac{1}{2}\log|LL'| \tag{113}$$

$$= \frac{1}{2}\log(|L|)^2 \tag{114}$$

$$= \log|L| \tag{115}$$

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

$$\frac{1}{2}\log|AZ'WZA + I| = \log\prod l_i \tag{116}$$

$$= \sum \log l_i \tag{117}$$

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

$$AZ'WZA = O\Lambda O' \tag{118}$$

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

$$AZ'WZA + I = O\Lambda O' + I = O\Lambda O + OO' = O(I + \Lambda)O' \tag{119}$$

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

$$|AZ'WZA + I| = 1 * \prod_{i=1}^{n}(1 + \lambda_i) * 1 \tag{120}$$

$$\Rightarrow \log|AZ'WZA + I| = \sum \log(1 + \lambda_i) \tag{121}$$

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.

9 Checks

9.1 Checking the MCLA finite differences

To check the function that calculates the MCLA $l_m(\theta)$, I use finite differences on the **?** example. To do this, I chose a value of $\theta = (\beta, \sigma)$ and a relatively small value of δ, where δ is a vector of length 2. We can check that the value and first derivative of the MCLA function are calculated correctly by making sure the following approximation holds

$$\nabla l_m(\theta) \cdot \delta \approx l_m(\theta + \delta) - l_m(\theta). \tag{122}$$

Then, we can check the first and second derivatives of the MCLA are calculated correctly by checking for the following approximation:

$$\nabla^2 l_m(\theta)\delta \approx \nabla l_m(\theta + \delta) - \nabla l_m(\theta) \tag{123}$$

9.2 Checking functions using the Booth and Hobert example

I also test the objective function by taking the **?** example and rewriting the functions for this specific example. I then compare the values produced by the check functions and the original functions. Functions checked this way are:

1. log of the data density (`el`)
2. log of the density for the random effects (both `distRand` and `distRandGeneral`)
3. the Monte Carlo likelihood approximation (`objfun`)

9.3 Checking a putative MLE using MCMC

Suppose $\hat{\theta}$ is claimed to be the MLE. For example, $\hat{\theta}$ could be the MCMLE. Ideally, we would like to check whether the true likelihood $l(\theta)$ achieves a local max at $\hat{\theta}$. Due to the intractible integral in the likelihood expression, the best we can do is make sure that $\nabla l_m(\theta)$ evaluated at $\hat{\theta}$ is very close to 0.

Suppose $u_k, k = 1, \ldots, m$ are sampled from importance sampling distribution $\tilde{f}(u_k)$. Recall that the gradient of the MCLA is:

$$\nabla l_m(\theta) = \frac{\sum_{k=1}^{m} \frac{\nabla f_\theta(y|u_k) f_\theta(u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^{m} \frac{f_\theta(u_k)}{\tilde{f}(u_k)}} \tag{124}$$

We can choose $\tilde{f}(u_k) = f_{\hat{\theta}}(u_k, y)$. If the putative MLE is truly the MLE, then $\theta = \hat{theta}$, which in turn implies that $\tilde{f}(u_k) = f_\theta(u_k, y)$. To be clear, in this check we no longer use a mixture of normals for $\tilde{f}(u_k)$. With the selected importance sampling distribution, each of the importance sampling weights is equal to 1 and the sum of the weights is m. Therefore, the gradient of the MCLA simplifies as follows:

$$\nabla l_m(\theta) = \frac{1}{m} \sum_{k=1}^{m} \frac{\nabla f_\theta(y|u_k) f_\theta(u_k)}{\tilde{f}(u_k, y)} \tag{125}$$

$$= \frac{1}{m} \sum_{k=1}^{m} \frac{\nabla f_\theta(u_k, y)}{\tilde{f}(u_k, y)} \tag{126}$$

$$= \frac{1}{m} \sum_{k=1}^{m} \nabla \log f_\theta(u_k, y) \tag{127}$$

This shows that the gradient of the MCLA is the average of the gradient of the complete data log likelihood, as long as $\hat{\theta}$ is truly an MLE. We can produce $u_k, k = 1, \ldots, m$, using Markov chain Monte Carlo. Using u as the variable, w run the Markov chain (perhaps `metrop` from the R package `mcmc`) on the complete data log likelihood. We then use these samples to calculate the gradient of the complete data log likelihood, which in turn calculates the gradient of the MCLA.

If we split the MCMC runs into batches, we can calculate the batch means of the MCLA gradient, the grand mean of the MCLA gradient, and the corresponding Monte Carlo standard error. If the putative MLE truly maximizes the likelihood, then the MCLA gradient's components should be close to 0. We can check that they are close enough to 0 by comparing them to the Monte Carlo standard error.

10 Confidence intervals

The user can implement this command to create confidence intervals after fitting a model *mod* using the main function. This is an S3 generic.The command would look like

```
confint(mod,parm,level=.95)
```

The only required argument would be the fitted model (the first argument). The third argument (the confidence level) has a default of .95.

If the second argument is omitted, confidence intervals would be created for all of the parameters. There are two options to calculate confidence intervals for a subset of the parameters. The user can provide either the names of the coefficients or a vector with length equal to the number of parameters (entries being 1 if they would like that intervals for that parameter and 0 otherwise). The code to do this is taken from "confint.lm" and is:

```
function (object, parm, level = 0.95, ...)
{
    cf <- coef(object)
    pnames <- names(cf)
    if (missing(parm))
        parm <- pnames
    else if (is.numeric(parm))
        parm <- pnames[parm]
        stopifnot(parm %in% pnames)

        rest of the code to actually do the confidence intervals goes here
        }
```

What this says: write down the coefficient names of the object we're given. If parm is missing, we're going to assume they want to create intervals for all the parameters. If parm is numeric (a vector of 0s and 1s), use that to select the parameter names we want and call that selection parm. Finally, stop the whole process if parm (the parameter names we want intervals for) don't appear in the model.

The form for a confidence interval is the point estimate plus or minus the standard error of the point estimate times some cutoff. The point estimate and the standard error are from the summary of the model. The reference distribution used for the cutoff is the standard normal. This will produce an asymptotic confidence interval.

The output would either be a vector of length 2 (if creating confidence interval for only one parameter) or a matrix with 2 columns (one for the lower bound of the confidence interval, one for the upper bond). The column names will be "$(100 - level)/2$" and "$50 + level/2$." (in the default case, this is"2.5%" and "97.5%").

The only potential hangup is when creating confidence intervals for the variance components $\nu_t, t = 1, ..., T$. We know that $\nu_t > 0$ for all $t = 1, ..., T$. We'll find this boundary again makes things complicated. I don't yet know how to deal with this.

To illustrate the problem, consider the scenario that the margin of error for ν_t is greater than the estimate of ν_t itself. It wouldn't make sense to produce a confidence interval with a negative lower bound. It could make sense to truncate the interval so that the lower bound starts at 0 and the upper bound remains untouched.

11 New Additions (Summer 2016)

11.1 New Family: Binomial

One year later, I'm adding binomial. For Binomial, the calculations change only in the cumulant function. They rely on one more number: the number of trials. The inputted response is usually a vector, but for binomial (with the exception of Bernoulli) we need 2 vectors cbinded (cbound?) together. The first column is the number of successes and the second is the number of failures. Summing across the row gives us the number of trials. I call this $ntrials$ from here on out.

The cumulant function and its first two derivatives with respect to the canonical parameter η_i are below.

$$c(\eta_i) = ntrials_i \log(1 + e^{\eta_i}) = \begin{cases} ntrials_i \log(1 + e^{\eta_i}) & \text{if } \eta_i \leq 0, \\ ntrials_i \ (\eta_i + \log(e^{-\eta_i} + 1)) & \text{if } \eta_i > 0, \end{cases} \tag{128}$$

$$c'(\eta_i) = \frac{e^{\eta_i}}{1 + e^{\eta_i}} = \frac{ntrials_i}{1 + e^{-\eta_i}} \tag{129}$$

$$c''(\eta_i) = \frac{e^{\eta_i}}{1 + e^{\eta_i}} - \frac{e^{2\eta_i}}{(1 + e^{\eta_i})^2} = \frac{ntrials_i}{1 + e^{-\eta_i}} \cdot \frac{1}{1 + e^{\eta_i}} \tag{130}$$

I will implement this change in steps. Game plan:

1. Without adding binomial yet, add the ntrials argument to all levels of code so that ntrials can be passed to the cumulant functions and their derivatives. At the top, I will have a variable to set ntrials = 1 (for now). As I do each layer, I will make sure I get all the same answers from my salamanders problem (or some other bernoulli problem).

2. Write the binomial cumulant in R. Do some checks with both η positive and negative.

   (a) Check to make sure this calculation is identical to that produced by bernoulli for some parameter value η_i and for $ntrials = 1$.

   (b) Check the sum of Bernoulli cumulants equals the binomial cumulant (for ntrials > 1).

   (c) Check finite differences.

   (d) Check the three calculations for $ntrials = 5$ (or some number other than 1) compared to my by-hand calculation.

3. Code it in C and check that the calculation is identical to that produced in R for $ntrials = 1$ and for $ntrials \neq 1$.

4. I will go to the top-most layer, right where the model matrices are being made. The new response vector y will be the first column of the user-provided response. The sum of the two

entries in the row will create the vector ntrials. I will check that the salamanders problem gives the same results (when specified as cbind(Mate, 1-Mate)).

5. I will add checks for the model matrix stage.
   (a) If the length of the dimension of the response is 2 dimensions (meaning we have both rows and columns), then the family must be binomial.glmm.
   (b) If the family is binomial.glmm, then the response must have 2 columns
6. I want to allow the response for bernoulli to be a single vector (rather than two cbinded vectors). Therefore, if the family is binomial.glmm and the length of the dimension of the response is 1 (meaning we just have a vector), then create the ntrials vector to be a vector of 1s. Test this with salamanders or some other problem to make sure it gives the same results.
7. Change PQL function to do binomial.

These changes will need to be done in both C and R. The checks will be in R. I'll need to add documentation for binomial. Bernoulli will be retained.

11.2 MCSE

Let γ_1 be defined as

$$\gamma_1 = \int f_\theta(u, y)\, du = f_\theta(y). \tag{131}$$

My thesis shows the Monte Carlo error of the gradient of the log likelihood at the MLE $\hat{\theta}$ is

$$V = \frac{1}{\gamma_1^2} \int \left[\nabla \log f_{\hat{\theta}}(u, y)\right] \left[\nabla \log f_{\hat{\theta}}(u, y)\right]^T \frac{f_{\hat{\theta}}(u, y)^2}{\tilde{f}(u)}\, du. \tag{132}$$

We can estimate this with the MCMLE $\hat{\theta}_m$:

$$\hat{V} = \frac{\frac{1}{m}\sum_{k=1}^m \left(\left[\nabla \log f_{\hat{\theta}_m}(u_k, y)\right] \left[\nabla \log f_{\hat{\theta}_m}(u_k, y)\right]^T \frac{f_{\hat{\theta}_m}(u_k, y)^2}{\tilde{f}(u_k)} \right)}{\left(\frac{1}{m}\sum_{k=1}^m \frac{f_{\hat{\theta}_m}(u_k, y)}{\tilde{f}(u_k)} \right)^2} \tag{133}$$

Let U be the Hessian of the likelihood evaluated at the MLE. This is estimated by

$$\hat{U} = \nabla^2 l_m(\hat{\theta}_m | y). \tag{134}$$

In other words, $\hat{U}$ is the MCLA Hessian evaluated at the MCMLE.

Then the MCSE of $\hat{\theta}_m$ is the square root of the diagonal elements of

$$\frac{1}{m} \hat{U}^{-1} \hat{V} \hat{U}^{-1} \tag{135}$$

under the regularity conditions listed in **?**.

The breakdown for programming MCSE is:

1. Get $\hat{U}$ from output's `likelihood.hessian`
2. Stably invert $\hat{U}$
3. Calculate $\hat{V}$
4. Finish calculation (result will be a vector of MCSE with length equal to number of parameters)
5. Output table

The output table will have a row for each parameter. The columns will be estimate (MCMLE), SE, MCSE. This will allow for easy comparison between SE and MCSE so that users can ensure their MCSE is small enough compared to the SE.

Note on calculating $\hat{U}$:the likelihood's Hessian is calculated for every call of objfunc. We can get this from the output's `likelihood.hessian`.

To do steps:

1. Remember how to do qr decomp (stably invert the matrix with qr.solve)
2. Calculate $\hat{V}$ and the rest of the MCSE calculation in R
3. Change the $\hat{V}$ calculation to C

11.2.1 Cholesky Decomposition for Matrix Inversion

Cholesky decomposition takes a symmetric, positive definite matrix and decomposes it into LL', where L is a lower triangular matrix and L' is the transpose of L (so it's upper triangular). Triangular matrices are easy to solve using `backsolve` in R. For example, if you have the hessian `hess` and want to get the variance covariance matrix, then do

```
cho <- chol(hess)
uinv <- backsolve(cho, diag(nrow(cho))
vcov <- uinv %*% t(uinv)
```

11.3 Revisit Summary Table

I just realized I have `solve()` for calculating the variance-covariance matrix. I need to do a QR decomposition or something more stable.

A MCLA calculations

Let $u_k, k = 1, \ldots, m$ be a sample from $\tilde{f}(u_k)$. The Monte Carlo log likelihood approximation is

$$l_m(\theta) = \log \frac{1}{m} \sum_{k=1}^{m} f_\theta(y|u_k) \frac{f_\theta(u_k)}{\tilde{f}(u_k)} \tag{136}$$

$$= \log \frac{1}{m} \sum_{k=1}^{m} \frac{f_\theta(y, u_k)}{\tilde{f}(u_k)}. \tag{137}$$

The gradient vector and Hessian matrix calculations depend on whether $\tilde{f}(u_k)$ contains θ. The calculations in A.2 are for $\tilde{f}(u_k)$ containing θ and the calculations in A.1 are for $\tilde{f}(u_k)$ not containing θ.

A.1 MCLA derivatives when $\tilde{f}$ independent of θ

When $\tilde{f}$ is independent of θ, the gradient vector of the MCLA with respect to θ is

$$\nabla l_m(\theta) = \frac{\sum_{k=1}^{m} \left(\nabla \log f_\theta(u_k, y) \frac{f_\theta(u_k, y)}{\tilde{f}(u_k)} \right)}{\sum_{k=1}^{m} \left(\frac{f_\theta(u_k, y)}{\tilde{f}(u_k)} \right)} \tag{138}$$

and the Hessian matrix of the MCLA is

$$\nabla^2 l_m(\theta) = \frac{\sum_{k=1}^{m} \left(\nabla^2 \log f_\theta(u_k, y) \frac{f_\theta(u_k, y)}{\tilde{f}(u_k)} \right)}{\sum_{k=1}^{m} \left(\frac{f_\theta(u_k, y)}{\tilde{f}(u_k)} \right)} - \nabla l_m(\theta)(\nabla l_m(\theta))' \tag{139}$$

$$+ \frac{\sum_{k=1}^{m} \left(\nabla \log f_\theta(u_k, y)(\nabla \log f_\theta(u_k, y))' \frac{f_\theta(u_k, y)}{\tilde{f}(u_k)} \right)}{\sum_{k=1}^{m} \left(\frac{f_\theta(u_k, y)}{\tilde{f}(u_k)} \right)}. \tag{140}$$

To reduce the risk of catastrophic cancellation, we can combine the last two terms of the Hessian:

$$\nabla^2 l_m(\theta) = \frac{\sum_{k=1}^m \nabla^2 \log f_\theta(y, u_k) \dfrac{f_\theta(y, u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \dfrac{f_\theta(y, u_k)}{\tilde{f}(u_k)}} \tag{141}$$

$$+ \frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y, u_k) - \nabla l_m(\theta)\right] \left[\nabla \log f_\theta(y, u_k) - \nabla l_m(\theta)\right]' \dfrac{f_\theta(y, u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \dfrac{f_\theta(y, u_k)}{\tilde{f}(u_k)}} \tag{142}$$

We are able to combine the last two terms because $\nabla l_m(\theta)$ is a weighted mean of $\nabla \log f_\theta(y, u_k) - \nabla \log \tilde{f}(u_k)$. Letting

$$Z = \nabla \log f_\theta(y, u_k) \tag{143}$$

we can use the following equality:

$$E(ZZ') - E(Z)E(Z)' = \left[E(Z - EZ)\right] \left[E(Z - EZ)\right]'. \tag{144}$$

A.2 MCLA derivatives when $\tilde{f}$ depends on θ

This section is still under construction.

When $\tilde{f}$ contains θ, the gradient of the MCLA is

$$\nabla l_m(\theta) = \frac{\nabla\left[\sum_{k=1}^{m}\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}\right]}{\sum_{k=1}^{m}\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{145}$$

$$= \frac{\sum_{k=1}^{m}\frac{\nabla f_\theta(y,u_k)}{\tilde{f}(u_k)} - \frac{f_\theta(y,u_k)\nabla\tilde{f}(u_k)}{\left(\tilde{f}(u_k)\right)^2}}{\sum_{k=1}^{m}\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{146}$$

$$= \frac{\sum_{k=1}^{m}\frac{\nabla f_\theta(y,u_k)}{f_\theta(y,u_k)}\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)} - \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}\frac{\nabla\tilde{f}(u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^{m}\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{147}$$

$$= \frac{\sum_{k=1}^{m}\nabla\log f_\theta(y,u_k)\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)} - \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}\nabla\log\tilde{f}(u_k)}{\sum_{k=1}^{m}\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{148}$$

$$= \frac{\sum_{k=1}^{m}\left[\nabla\log f_\theta(y,u_k) - \nabla\log\tilde{f}(u_k)\right]\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^{m}\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{149}$$

Then the Hessian of the MCLA is

$$\nabla^2 l_m(\theta) = \nabla \left[\frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right] \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \right] \tag{150}$$

$$= \frac{\sum_{k=1}^m \left[\nabla^2 \log f_\theta(y,u_k) - \nabla^2 \log \tilde{f}(u_k) \right] \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{151}$$

$$+ \frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right] \nabla \left[\frac{f_\theta(y,u_k)}{\tilde{f}(u_k)} \right]}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{152}$$

$$- \left[\frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right] \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\left(\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)} \right)^2} \right] \nabla \left[\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)} \right] \tag{153}$$

$$= \frac{\sum_{k=1}^m \left[\nabla^2 \log f_\theta(y,u_k) - \nabla^2 \log \tilde{f}(u_k) \right] \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{154}$$

$$+ \frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right] \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right]' \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{155}$$

$$- \left[\frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right] \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\left(\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)} \right)^2} \right] \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right]' \tag{156}$$

$$= \frac{\sum_{k=1}^m \left[\nabla^2 \log f_\theta(y,u_k) - \nabla^2 \log \tilde{f}(u_k) \right] \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{157}$$

$$+ \frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right] \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \right]' \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{158}$$

$$- \left[\nabla l_m(\theta) \right] \left[\nabla l_m(\theta) \right]' \tag{159}$$

To reduce the risk of catastrophic cancellation, we can combine the last two terms of the Hessian:

$$\nabla^2 l_m(\theta) = \frac{\sum_{k=1}^m \left[\nabla^2 \log f_\theta(y,u_k) - \nabla^2 \log \tilde{f}(u_k)\right] \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{160}$$

$$+ \frac{\sum_{k=1}^m \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) - \nabla l_m(\theta)\right] \left[\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) - \nabla l_m(\theta)\right]' \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \frac{f_\theta(y,u_k)}{\tilde{f}(u_k)}} \tag{161}$$

We are able to combine the last two terms because $\nabla l_m(\theta)$ is a weighted mean of $\nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k)$. Letting

$$Z = \nabla \log f_\theta(y,u_k) - \nabla \log \tilde{f}(u_k) \tag{162}$$

we can use the following equality:

$$E(ZZ') - E(Z)E(Z)' = [E(Z - EZ)]\,[E(Z - EZ)]'. \tag{163}$$

B Central Limit Theorem for MCLA

In this section, we focus on the gradient of the MCLA because the trust criterion for finding the maximum is based on the gradient when $\tilde{f}(u)$ is independent of θ. Define

$$\gamma_1 = \int f_\theta(u,y)du \tag{164}$$

$$\gamma_2 = \int \tilde{f}(u)du. \tag{165}$$

Recall the calculation for the MCLA gradient originally stated in (4):

$$\nabla l_m(\theta) = \frac{\sum_{k=1}^m \nabla \log f_\theta(u_k,y) \frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}}{\sum_{k=1}^m \left(\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\right)} \tag{166}$$

Note that both the numerator and denominator are sample means. By the law of large numbers,

the numerators and denominator each converge to their true means. That is,

$$\frac{1}{m}\sum_{k=1}^{m}\nabla \log f_\theta(u_k,y)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} \to E_{\tilde{f}}\left[\nabla \log f_\theta(u,y)\frac{f_\theta(u,y)}{\tilde{f}(u)}\right] \tag{167}$$

$$= \int \nabla \log f_\theta(u,y)\frac{f_\theta(u,y)}{\tilde{f}(u)}\frac{\tilde{f}(u)}{\gamma_2}du \tag{168}$$

$$= \frac{\gamma_1}{\gamma_2}\int \nabla \log f_\theta(u,y)\frac{f_\theta(u,y)}{\gamma_1}du \tag{169}$$

$$= \frac{\gamma_1}{\gamma_2}E_f\left[\nabla \log f_\theta(u,y)\right] \tag{170}$$

and

$$\frac{1}{m}\sum_{k=1}^{m}\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} \to E_{\tilde{f}}\left[\frac{f_\theta(u,y)}{\tilde{f}(u)}\right] \tag{171}$$

$$= \int \frac{f_\theta(u,y)}{\tilde{f}(u)}\frac{\tilde{f}(u)}{\gamma_2}du \tag{172}$$

$$= \frac{\gamma_1}{\gamma_2}\int \frac{f_\theta(u,y)}{\gamma_2}du \tag{173}$$

$$= \frac{\gamma_1}{\gamma_2} \tag{174}$$

Then, by Slutsky's theorem,

$$\nabla l_m(\theta) \to \frac{\frac{\gamma_1}{\gamma_2}E_f\left[\nabla \log f_\theta(u,y)\right]}{\frac{\gamma_1}{\gamma_2}} \tag{175}$$

$$= E_f\left[\nabla \log f_\theta(u,y)\right] \tag{176}$$

$$= \nabla l(\theta) \tag{177}$$

In addition to a law of large numbers, we would like a Central Limit Theorem for $\nabla l_m(\theta)$. In other

words, the quantity

$$\sqrt{m}\left[\frac{\frac{1}{m}\sum_{k=1}^{m}\nabla\log f_\theta(u_k,y)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}}{\frac{1}{m}\sum_{k=1}^{m}\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}}-\nabla l(\theta)\right] \tag{178}$$

$$=\frac{\frac{1}{\sqrt{m}}\sum_{k=1}^{m}\nabla\log f_\theta(u_k,y)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}}{\frac{1}{m}\sum_{k=1}^{m}\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}}-\sqrt{m}\,\nabla l(\theta) \tag{179}$$

$$=\frac{\frac{1}{\sqrt{m}}\sum_{k=1}^{m}\nabla\log f_\theta(u_k,y)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}-\nabla l(\theta)\frac{1}{\sqrt{m}}\sum_{k=1}^{m}\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}}{\frac{1}{m}\sum_{k=1}^{m}\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}} \tag{180}$$

will have a normal distribution if and only if the variances of

$$\sum_{k=1}^{m}\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} \tag{181}$$

and

$$\sum_{k=1}^{m}\nabla\log f_\theta(u_k,y)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} \tag{182}$$

are finite. First, we want to show that

$$\sum_{k=1}^{m}\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} \tag{183}$$

has finite variance. This is true if and only if

$$\int\frac{f_\theta(u,y)^2}{\tilde{f}(u)}du<\infty. \tag{184}$$

We see:

$$\int\frac{f_\theta(u,y)^2}{\tilde{f}(u)}du=\int\frac{f_\theta(y|u)^2f_\theta(u)^2}{\tilde{f}(u)}du \tag{185}$$

$$\leq\int\frac{f_\theta(y|u)^2f_\theta(u)^2}{p_1\grave{f}(u|0,D^*)}du \tag{186}$$

$$\propto\frac{1}{p_1}\int\frac{\exp(-u'D^{-1}u)\,f_\theta(y|u)^2}{[1+u'D^*u/\zeta]^{-(\zeta+q)/2}}du \tag{187}$$

Since

$$f_\theta(y|u)^2=\left(e^{y'\eta-c(\eta)}\right)^2=e^{2y'\eta-2c(\eta)}, \tag{188}$$

$$\int \frac{f_\theta(u,y)^2}{\tilde{f}(u)}du \propto \frac{1}{p_1}\int \frac{\exp(-u'D^{-1}u)\,\exp(2y'\eta)}{[1+u'D^*u/\zeta]^{-(\zeta+q)/2}\,\exp(2c(\eta))}du \tag{189}$$

$$= \frac{1}{p_1}\int \frac{[1+u'D^*u/\zeta]^{(\zeta+q)/2}\,\exp(2y'\eta)}{\exp(u'D^{-1}u)\,\exp(2c(\eta))}du \tag{190}$$

$$= \frac{1}{p_1}\int \frac{[1+u'D^*u/\zeta]^{(\zeta+q)/2}\,\exp(2y'Zu)}{\exp(u'D^{-1}u)\,\exp(2c(\eta))}du \tag{191}$$

When $y|u \sim$ Bernoulli, then

$$c(\eta) = \log(1+e^\eta) \Rightarrow \exp(2c(\eta)) = \exp(2\log(1+e^\eta)) \tag{192}$$

$$= (\exp(\log(1+e^\eta)))^2 \tag{193}$$

$$= (1+e^\eta)^2 \tag{194}$$

and

$$\int \frac{f_\theta(u,y)^2}{\tilde{f}(u)}du \propto \frac{1}{p_1}\int \frac{[1+u'D^*u/\zeta]^{(\zeta+q)/2}\,\exp(2y'Zu)}{\exp(u'D^{-1}u)\,(1+e^\eta)^2}du, \tag{195}$$

which converges since $\exp(u'D^{-1}u$ grows faster than any term in the numerator of the integrand.

On the other hand, when $y|u \sim$ Poisson, then

$$c(\eta) = \exp(\eta) \Rightarrow \exp(2c(\eta)) = (\exp(\exp(\eta)))^2 \tag{196}$$

and

$$\int \frac{f_\theta(u,y)^2}{\tilde{f}(u)}du \propto \frac{1}{p_1}\int \frac{[1+u'D^*u/\zeta]^{(\zeta+q)/2}\,\exp(2y'Zu)}{\exp(u'D^{-1}u)\,(\exp(\exp(\eta)))^2}du, \tag{197}$$

which converges since $(\exp(\exp(\eta)))^2$ grows faster than any term in the numerator of the integrand.

Therefore, whether the data are Bernoulli or Poisson distributed, the importance sampling weights

$$\sum_{k=1}^{m} \frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} \tag{198}$$

have finite variance.

Now, we want to show that

$$\sum_{k=1}^{m} \nabla \log f_\theta(u_k,y)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} = \sum_{k=1}^{m} \nabla \log f_\theta(y|u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} + \sum_{k=1}^{m} \nabla \log f_\theta(u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)} \tag{199}$$

has finite variance. We need to show the following are finite:

$$\operatorname{Var}\left(\sum_{k=1}^{m}\nabla\log f_\theta(y|u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\right) \tag{200}$$

$$\operatorname{Var}\left(\sum_{k=1}^{m}\nabla\log f_\theta(u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\right) \tag{201}$$

$$\operatorname{Cov}\left(\sum_{k=1}^{m}\nabla\log f_\theta(y|u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\,,\sum_{k=1}^{m}\nabla\log f_\theta(u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\right). \tag{202}$$

That is, we want to show the existence of the following:

$$\int\frac{\left[\nabla\log f_\theta(y|u)\right]^2\left[f_\theta(u,y)\right]^2}{\tilde{f}(u)}du \tag{203}$$

$$\int\frac{\left[\nabla\log f_\theta(u)\right]^2\left[f_\theta(u,y)\right]^2}{\tilde{f}(u)}du \tag{204}$$

$$\int\frac{\nabla\log f_\theta(u)\nabla\log f_\theta(y|u)\left[f_\theta(u,y)\right]^2}{\tilde{f}(u)}du \tag{205}$$

First, we start with (203):

$$\int\frac{\left[\nabla\log f_\theta(y|u)\right]^2\left[f_\theta(u,y)\right]^2}{\tilde{f}(u)}du = \int\frac{\left[\nabla\log f_\theta(y|u)\right]^2\left[f_\theta(y|u)\right]^2\left[f_\theta(u)\right]^2}{\tilde{f}(u)}du \tag{206}$$

$$\leq\int\frac{\left[\nabla\log f_\theta(y|u)\right]^2\left[f_\theta(y|u)\right]^2\left[f_\theta(u)\right]^2}{p_1\hat{f}(u|0,D^*)}du \tag{207}$$

$$\propto\int\frac{\left[\nabla\log f_\theta(y|u)\right]^2\left[f_\theta(y|u)\right]^2\exp(-u'D^{-1}u)}{p_1\left[1+u'D^*u/\zeta\right]^{-(\zeta+q)/2}}du \tag{208}$$

$$=\int\frac{\left[\nabla\log f_\theta(y|u)\right]^2\left[f_\theta(y|u)\right]^2\left[1+u'D^*u/\zeta\right]^{(\zeta+q)/2}}{p_1\exp(u'D^{-1}u)}du \tag{209}$$

$$=\int\frac{\left[\nabla\log f_\theta(y|u)\right]^2\exp(2y'(X\beta+Zu))\left[1+u'D^*u/\zeta\right]^{(\zeta+q)/2}}{p_1\exp(2c(\eta))\exp(u'D^{-1}u)}du \tag{210}$$

By the Cauchy-Schwartz inequality, the integral in (210) exists if

$$\int\frac{\left[\dfrac{\partial}{\partial\beta}\log f_\theta(y|u)\right]^2\exp(2y'(X\beta+Zu))\left[1+u'D^*u/\zeta\right]^{(\zeta+q)/2}}{p_1\exp(2c(\eta))\exp(u'D^{-1}u)}du$$

$$\propto\int\frac{\left[X'c'(\eta)\right]^2\exp(2y'(X\beta+Zu))\left[1+u'D^*u/\zeta\right]^{(\zeta+q)/2}}{p_1\exp(2c(\eta))\exp(u'D^{-1}u)}du \tag{211}$$

exists, where $c'(\eta)$ is a vector with components $c'(\eta_i)$. This will converge as long as no term in the numerator of the integrand grows more quickly than $\exp(u'D^{-1}u$. The second and third terms grow less quickly than $\exp(u'D^{-1}u$, so $[c'(\eta)]^2$ is the only term to check. If $y|u \sim$ Bernoulli, then

$$c'(\eta_i) = \frac{\exp(\eta_i)}{1+\exp(\eta_i)}, \tag{212}$$

which does not grow more quickly than $\exp(u'D^{-1}u)$. If $y|u \sim$ Poisson, then

$$c'(\eta_i) = e^{\eta_i}, \tag{213}$$

which does not grow more quickly than $\exp(u'D^{-1}u)$. Therefore, (203) exists and is finite, and

$$\text{Var}\left(\sum_{k=1}^{m} \nabla \log f_\theta(y|u_k) \frac{f_\theta(u_k, y)}{\tilde{f}(u_k)}\right) < \infty. \tag{214}$$

is finite.

Next, we move on to showing the existence of (204).

$$\int \frac{[\nabla \log f_\theta(u)]^2 [f_\theta(u,y)]^2}{\tilde{f}(u)} du = \int \frac{[\nabla \log f_\theta(u)]^2 [f_\theta(y|u)]^2 [f_\theta(u)]^2}{\tilde{f}(u)} du \tag{215}$$

$$\leq \int \frac{[\nabla \log f_\theta(u)]^2 [f_\theta(y|u)]^2 [f_\theta(u)]^2}{p_1 \grave{f}(u|0, D^*)} du \tag{216}$$

By the Cauchy-Schwartz inequality, the above integral exists as long as

$$\int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right]^2 [f_\theta(y|u)]^2 [f_\theta(u)]^2}{p_1 \grave{f}(u|0, D^*)} du < \infty \tag{217}$$

for every $t = 1, \ldots, T$. We see

$$\int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right]^2 [f_\theta(y|u)]^2 [f_\theta(u)]^2}{p_1 \grave{f}(u|0, D^*)} du \tag{218}$$

$$= \int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right]^2 \exp(2y'(X\beta + Zu)) [f_\theta(u)]^2}{p_1 \exp(2c(\eta)) \grave{f}(u|0, D^*)} du \tag{219}$$

$$\propto \int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right]^2 \exp(2y'(X\beta + Zu)) \ [1 + u'D^*u/\zeta]^{(\zeta+q)/2}}{p_1 \exp(2c(\eta)) \ \exp(u'D^{-1}u)} du \tag{220}$$

$$= \int \frac{\left[-\frac{q_t}{2\nu_t} + \frac{u_t' u_t}{2\nu_t^2}\right]^2 \exp(2y'(X\beta + Zu)) \ [1 + u'D^*u/\zeta]^{(\zeta+q)/2}}{p_1 \exp(2c(\eta)) \ \exp(u'D^{-1}u)} du \tag{221}$$

Because every term in the numerator of the integrand grows less quickly than $\exp(u'D^{-1}u)$, the integral exists. Therefore, (204) exists and

$$\text{Var}\left(\sum_{k=1}^{m} \nabla \log f_\theta(u_k) \frac{f_\theta(u_k, y)}{\tilde{f}(u_k)}\right) < \infty. \tag{222}$$

Lastly, we want to show the existence of (205). We see

$$\int \frac{\nabla \log f_\theta(u) \nabla \log f_\theta(y|u) \left[f_\theta(u,y)\right]^2}{\tilde{f}(u)} du = \int \frac{\nabla \log f_\theta(u) \nabla \log f_\theta(y|u) \left[f_\theta(y|u)\right]^2 \left[f_\theta(u)\right]^2}{\tilde{f}(u)} du \tag{223}$$

$$\leq \int \frac{\nabla \log f_\theta(u) \nabla \log f_\theta(y|u) \left[f_\theta(y|u)\right]^2 \left[f_\theta(u)\right]^2}{p_1 \grave{f}(u|0, D^*)} du \tag{224}$$

Again, we use the Cauchy-Schwartz inequality to check the existence of (225). If, for every $t = 1, \ldots, T$,

$$\int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right] \left[\frac{\partial}{\partial \beta} \log f_\theta(y|u)\right] \left[f_\theta(y|u)\right]^2 \left[f_\theta(u)\right]^2}{p_1 \grave{f}(u|0, D^*)} du \tag{225}$$

exists, then (225) also exists. Continuing, we see

$$\int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right] \left[\frac{\partial}{\partial \beta} \log f_\theta(y|u)\right] \left[f_\theta(y|u)\right]^2 \left[f_\theta(u)\right]^2}{p_1 \grave{f}(u|0, D^*)} du \tag{226}$$

$$\propto \int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right] \left[\frac{\partial}{\partial \beta} \log f_\theta(y|u)\right] \left[f_\theta(y|u)\right]^2}{p_1 \exp(u'D^{-1}u) \grave{f}(u|0, D^*)} du \tag{227}$$

$$\propto \int \frac{\left[\frac{\partial}{\partial \nu_t} \log f_\theta(u)\right] \left[\frac{\partial}{\partial \beta} \log f_\theta(y|u)\right] \left[f_\theta(y|u)\right]^2 \left[1 + u'D^*u/\zeta\right]^{(\zeta+q)/2}}{p_1 \exp(u'D^{-1}u)} du \tag{228}$$

$$\propto \int \frac{\left[-\frac{q_t}{2\nu_t} + \frac{u_t'u_t}{2\nu_t^2}\right] \left[X'(y - c'(\eta))\right] \left[f_\theta(y|u)\right]^2 \left[1 + u'D^*u/\zeta\right]^{(\zeta+q)/2}}{p_1 \exp(u'D^{-1}u)} du \tag{229}$$

$$= \int \frac{\left[-\frac{q_t}{2\nu_t} + \frac{u_t'u_t}{2\nu_t^2}\right] \left[X'(y - c'(\eta))\right] \exp(2y'(X\beta + Zu)) \left[1 + u'D^*u/\zeta\right]^{(\zeta+q)/2}}{p_1 \exp(u'D^{-1}u) \exp(2c(\eta))} du. \tag{230}$$

We have already shown that $-c'(\eta)$ does not grow more quickly than $\exp(u'D^{-1}u)$. The other terms in the numerator of the integrand shown in (230) also grow more slowly than $\exp(u'D^{-1}u)$.

We conclude that (230) exists, which implies that (205) exists and

$$\mathrm{Cov}\left(\sum_{k=1}^{m}\nabla\log f_\theta(y|u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\,,\,\sum_{k=1}^{m}\nabla\log f_\theta(u_k)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\right)<\infty. \tag{231}$$

This shows that

$$\mathrm{Var}\left(\sum_{k=1}^{m}\nabla\log f_\theta(u_k,y)\frac{f_\theta(u_k,y)}{\tilde{f}(u_k)}\right)<\infty. \tag{232}$$

Therefore, the gradient of the MCLA has finite variance and a central limit theorem.