Checking the quality of Laplace-approximate inference

Helen Ogden, University of Southampton

ADMB/TMB developers' meeting 2017

Example 1: a two-level model

Have binary observations y_i which are clustered: each i belongs to a cluster $c(i)$.

Example 1: a two-level model

Have binary observations y_i which are clustered: each i belongs to a cluster $c(i)$.

Model

$$Pr(Y_i = 1|\eta_i) = \text{logit}^{-1}(\eta_i)$$

Example 1: a two-level model

Have binary observations y_i which are clustered: each i belongs to a cluster $c(i)$.

Model

$$Pr(Y_i = 1|\eta_i) = \text{logit}^{-1}(\eta_i)$$

and

$$\eta_i = \alpha + \beta x_i + \sigma u_{\mathrm{c}(i)}$$

where $u_j \sim N(0, 1)$.

Example 1: a two-level model

Have binary observations y_i which are clustered: each i belongs to a cluster $c(i)$.

Model

$$Pr(Y_i = 1|\eta_i) = \mathrm{logit}^{-1}(\eta_i)$$

and

$$\eta_i = \alpha + \beta x_i + \sigma u_{\mathrm{c}(i)}$$

where $u_j \sim N(0, 1)$.

Want to do inference on $\theta = (\alpha, \beta, \sigma)$.

Example 1: a two-level model

```
library(lme4)
```

```
glmer(response ~ covariate + (1 | cluster), data = two_level,
      family = binomial)
```

Example 1: a two-level model

```r
library(lme4)
```

```r
glmer(response ~ covariate + (1 | cluster), data = two_level,
      family = binomial)
```

```
## Generalized linear mixed model fit by maximum
##   likelihood (Laplace Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: response ~ covariate + (1 | cluster)
##    Data: two_level
##      AIC      BIC   logLik deviance df.resid
## 137.8656 145.6811 -65.9328 131.8656       97
## Random effects:
##  Groups  Name        Std.Dev.
##  cluster (Intercept) 0.7475
## Number of obs: 100, groups:  cluster, 50
## Fixed Effects:
## (Intercept)    covariate
##      0.6521      -1.1575
```

The likelihood

Write

$$f_y(y_i|\theta, u_{c(i)}) = Pr(Y_i = y_i|\eta_i = \alpha + \beta x_i + \sigma u_{c(i)})$$

Then

$$L(\theta|\mathbf{y}) = \int_{\mathbb{R}^n} \prod_{i=1}^{m} f_y(y_i|\theta, u_{c(i)}) \prod_{j=1}^{n} \phi(u_j) d\mathbf{u}$$

An n-dimensional integral.

Laplace approximation to the likelihood

Write

$$L(\theta) = \int_{\mathbb{R}^n} \exp\{-g(\mathbf{u};\theta)\} d\mathbf{u}$$

where

$$-g(\mathbf{u};\theta) = \sum_{i=1}^{m} \log f_y(y_i|\theta, u_{c(i)}) + \sum_{j=1}^{n} \log \phi(u_j).$$

Laplace approximation to the likelihood

Write

$$L(\theta) = \int_{\mathbb{R}^n} \exp\{-g(\mathbf{u};\theta)\} d\mathbf{u}$$

where

$$-g(\mathbf{u};\theta) = \sum_{i=1}^{m} \log f_y(y_i|\theta, u_{c(i)}) + \sum_{j=1}^{n} \log \phi(u_j).$$

Idea: find a normal approximation to the integrand.

$$\exp\{-\tilde{g}(\mathbf{u};\theta)\} = c_\theta \phi_n(\mathbf{b}; \mu_\theta, \Sigma_\theta)$$

μ_θ is the minimizer of $g(.;\theta)$ over $\mathbf{u}$,

Laplace approximation to the likelihood

Write

$$L(\theta) = \int_{\mathbb{R}^n} \exp\{-g(\mathbf{u};\theta)\} d\mathbf{u}$$

where

$$-g(\mathbf{u};\theta) = \sum_{i=1}^{m} \log f_y(y_i|\theta, u_{c(i)}) + \sum_{j=1}^{n} \log \phi(u_j).$$

Idea: find a normal approximation to the integrand.

$$\exp\{-\tilde{g}(\mathbf{u};\theta)\} = c_\theta \phi_n(\mathbf{b};\mu_\theta, \Sigma_\theta)$$

μ_θ is the minimizer of $g(.;\theta)$ over $\mathbf{u}$, Σ_θ is found by using curvature of $g(.;\theta)$ at μ_θ,

Laplace approximation to the likelihood

Write

$$L(\theta) = \int_{\mathbb{R}^n} \exp\{-g(\mathbf{u};\theta)\} d\mathbf{u}$$

where

$$-g(\mathbf{u};\theta) = \sum_{i=1}^{m} \log f_y(y_i|\theta, u_{c(i)}) + \sum_{j=1}^{n} \log \phi(u_j).$$

Idea: find a normal approximation to the integrand.

$$\exp\{-\tilde{g}(\mathbf{u};\theta)\} = c_\theta \phi_n(\mathbf{b};\mu_\theta, \Sigma_\theta)$$

μ_θ is the minimizer of $g(.;\theta)$ over $\mathbf{u}$, Σ_θ is found by using curvature of $g(.;\theta)$ at μ_θ, and c_θ is chosen so that $\tilde{g}(\mu_\theta;\theta) = g(\mu_\theta;\theta)$.

Simplification in the two-level model

Recall

$$L(\theta|\mathbf{y}) = \int_{\mathbb{R}^n} \prod_{i=1}^{m} f_y(y_i|\theta, u_{c(i)}) \prod_{j=1}^{n} \phi(u_j) d\mathbf{u}$$

in the two-level model.

Simplification in the two-level model

Recall

$$L(\theta|\mathbf{y}) = \int_{\mathbb{R}^n} \prod_{i=1}^{m} f_y(y_i|\theta, u_{c(i)}) \prod_{j=1}^{n} \phi(u_j) d\mathbf{u}$$

in the two-level model.

But

$$L(\theta|\mathbf{y}) = \prod_{j=1}^{n} \int_{-\infty}^{\infty} \prod_{i:c(i)=j} f_y(y_i|\theta, u_j)\phi(u_j) du_j$$

so only need to compute one-dimensional integrals.

Example 1: a two-level model

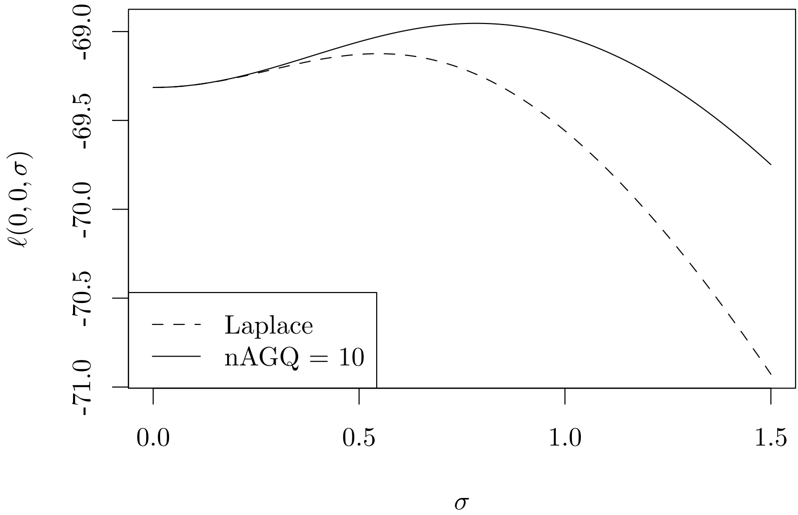
```r
glmer(response ~ covariate + (1 | cluster), data = two_level,
      family = binomial, nAGQ = 10)
```

Example 1: a two-level model

```r
glmer(response ~ covariate + (1 | cluster), data = two_level,
      family = binomial, nAGQ = 10)
```

```
## Generalized linear mixed model fit by maximum
##   likelihood (Adaptive Gauss-Hermite
##   Quadrature, nAGQ = 10) [glmerMod]
##  Family: binomial  ( logit )
## Formula: response ~ covariate + (1 | cluster)
##    Data: two_level
##      AIC      BIC   logLik deviance df.resid
## 137.2254 145.0409 -65.6127 131.2254       97
## Random effects:
##  Groups  Name        Std.Dev.
##  cluster (Intercept) 1.041
## Number of obs: 100, groups:  cluster, 50
## Fixed Effects:
## (Intercept)    covariate
##      0.7167      -1.2734
```

Comparing approximations to the loglikelihood



Example 2: a three-level model

Each cluster j is itself contained within larger group $g(c)$.

Example 2: a three-level model

Each cluster j is itself contained within larger group $g(c)$.

Have

$$\eta_i = \alpha + \beta x_i + \sigma_c u_{c(i)} + \sigma_g v_{g(c(i))}$$

where each $u_j, v_j \sim N(0,1)$.

Do inference on $\theta = (\alpha, \beta, \sigma_c, \sigma_g)$

Example 2: a three-level model

```r
glmer(response ~ covariate + (1 | cluster) + (1 | group),
      data = three_level, family = binomial)
```

Example 2: a three-level model

```r
glmer(response ~ covariate + (1 | cluster) + (1 | group),
      data = three_level, family = binomial)
```

```
## Generalized linear mixed model fit by maximum
##   likelihood (Laplace Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## response ~ covariate + (1 | cluster) + (1 | group)
##    Data: three_level
##       AIC       BIC    logLik  deviance  df.resid
##  283.4225  296.6157 -137.7112  275.4225       196
## Random effects:
##  Groups  Name        Std.Dev.
##  cluster (Intercept) 0.3576
##  group   (Intercept) 0.4257
## Number of obs: 200, groups:
## cluster, 100; group, 50
## Fixed Effects:
## (Intercept)    covariate
##     -0.1908       0.1198
```

Example 2: a three-level model

```
glmer(response ~ covariate + (1 | cluster) + (1 | group),
      data = three_level, family = binomial, nAGQ = 10)
```

Example 2: a three-level model

```
glmer(response ~ covariate + (1 | cluster) + (1 | group),
      data = three_level, family = binomial, nAGQ = 10)
```

```
## Error in updateGlmerDevfun(devfun, glmod$reTrms, nAGQ = nAGQ): nAGQ
```

The `glmmsr` package

Fits a GLMM with a call of the form

```
glmm(formula, data, family, method, ...)
```

`method` is the method used to approximate the likelihood.

The glmmsr package

Fits a GLMM with a call of the form

```
glmm(formula, data, family, method, ...)
```

`method` is the method used to approximate the likelihood.

There is no default for method.

The current state of `glmmsr`

- a choice of four methods for approximating the likelihood (`"Laplace"`, `"AGQ"`, `"SR"` and `"IS"`)

The current state of `glmmsr`

- a choice of four methods for approximating the likelihood (`"Laplace"`, `"AGQ"`, `"SR"` and `"IS"`)
- no default method

The current state of `glmmsr`

- a choice of four methods for approximating the likelihood (`"Laplace"`, `"AGQ"`, `"SR"` and `"IS"`)
- no default method
- some guidance on choice of method provided in vignette

The current state of `glmmsr`

- a choice of four methods for approximating the likelihood (`"Laplace"`, `"AGQ"`, `"SR"` and `"IS"`)
- no default method
- some guidance on choice of method provided in vignette

Pros

- allows easy experimentation with different methods for approximating the likelihood
- forces a user to think about issue of likelihood approximation

The current state of `glmmsr`

- a choice of four methods for approximating the likelihood (`"Laplace"`, `"AGQ"`, `"SR"` and `"IS"`)
- no default method
- some guidance on choice of method provided in vignette

Pros

- allows easy experimentation with different methods for approximating the likelihood
- forces a user to think about issue of likelihood approximation

Cons

- choosing an appropriate `method` may be difficult
- no warnings given by default if the choice of `method` gives a poor approximation to the likelihood

Ultimate aim: Sensible defaults, useful warnings

Ideally, would like different levels of defaults:

- by default, choose cheapest method which will give inference "reasonably close" to inference with the exact likelihood

Ultimate aim: Sensible defaults, useful warnings

Ideally, would like different levels of defaults:

- by default, choose cheapest method which will give inference "reasonably close" to inference with the exact likelihood
- for more control, user could specify a tolerance on how close the inference must be to inference with exact likelihood

Ultimate aim: Sensible defaults, useful warnings

Ideally, would like different levels of defaults:

- by default, choose cheapest method which will give inference "reasonably close" to inference with the exact likelihood
- for more control, user could specify a tolerance on how close the inference must be to inference with exact likelihood
- still allow full control over method

Ultimate aim: Sensible defaults, useful warnings

Ideally, would like different levels of defaults:

- by default, choose cheapest method which will give inference "reasonably close" to inference with the exact likelihood
- for more control, user could specify a tolerance on how close the inference must be to inference with exact likelihood
- still allow full control over method
- give a warning if method used might give inference far from inference with exact likelihood.

First steps: when is the Laplace approximation good enough?

Aim: given a model and data, determine quickly whether or not inference using the Laplace approximation to the likelihood is sufficiently close to inference using the exact likelihood.

The development version of `glmmsr`, available at `https://github.com/heogden/glmmsr/` includes the option to run some checks of the quality of the Laplace approximation.

First steps: when is the Laplace approximation good enough?

Aim: given a model and data, determine quickly whether or not inference using the Laplace approximation to the likelihood is sufficiently close to inference using the exact likelihood.

The development version of `glmmsr`, available at `https://github.com/heogden/glmmsr/` includes the option to run some checks of the quality of the Laplace approximation.

Previous and ongoing work: asymptotic view of this problem.

Ogden, H. E. (2017). On asymptotic validity of naive inference with an approximate likelihood. Biometrika, (February), 153–164. `https://doi.org/10.1093/biomet/asx002`

First steps: when is the Laplace approximation good enough?

Go back to our two-level model

```
fit_Laplace <- glmm(response ~ covariate + (1 | cluster),
                    data = two_level,
                    family = binomial, method = "Laplace",
                    control = list(check_Laplace = TRUE))
```

```
## Fitting the model. done.
```

```
## Warning: Inference using the first-order Laplace
## approximation may be unreliable in this case
```

With more observations on each cluster

Let's check what happens on another two-level model, with 10 items in each cluster, rather than 2.

```
fit_Laplace_10 <- glmm(response ~ covariate + (1 | cluster),
                       data = two_level_10,
                       family = binomial, method = "Laplace",
                       control = list(check_Laplace = TRUE))
```

```
## Fitting the model. done.
```

No warnings.

Examining the inference with the Laplace approximation

```r
summary(fit_Laplace_10)
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Laplace approximation, order  1
##
## Family: binomial ( logit )
## Formula: response ~ covariate + (1 | cluster)
##
## Random effects:
##  Groups  Name        Estimate Std.Error
##  cluster (Intercept) 0.6844   0.1529
## Number of obs: 500, groups: cluster, 50;
##
## Fixed effects:
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)   0.2255     0.1627   1.386 0.1658851
## covariate    -0.7048     0.1975   3.569 0.0003579
```

Fitting the model with AGQ

```r
fit_exact_10 <- glmm(response ~ covariate + (1 | cluster),
                     data = two_level_10,
                     family = binomial, method = "AGQ",
                     control = list(nAGQ = 20))
```

```
## Fitting the model. done.
```

Examining the inference with the AGQ approximation

```r
summary(fit_exact_10)
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Adaptive Gaussian Quadrature with 20 point
##
## Family: binomial ( logit )
## Formula: response ~ covariate + (1 | cluster)
##
## Random effects:
##  Groups  Name        Estimate Std.Error
##  cluster (Intercept) 0.7041   0.1555
## Number of obs: 500, groups: cluster, 50;
##
## Fixed effects:
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)   0.2249     0.1645   1.367 0.1716733
## covariate    -0.7037     0.1976   3.561 0.0003699
```

How do the warnings work?

Suppose that the exact likelihood was actually available, but very costly to compute relative to the Laplace approximation. How would we check how "close" the inference found using the Laplace approximation is to the inference found using the exact likelihood?

A simplification of the model

To make it easier to visualize what's going on, let's drop the intercept from the model, so that there are only two parameters.

```
fit_Laplace <- glmm(response ~ 0 + covariate + (1 | cluster),
                    data = two_level,
                    family = binomial, method = "Laplace",
                    control = list(check_Laplace = TRUE))
```

```
## Fitting the model. done.
```

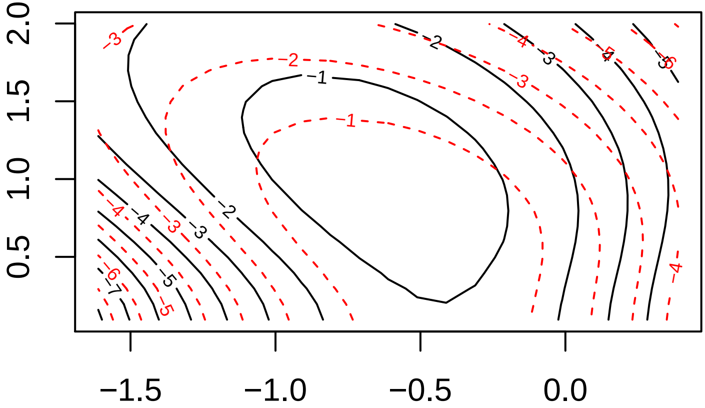
```
## Warning: Inference using the first-order Laplace
## approximation may be unreliable in this case
```

(Approximate) posterior distribution

Putting a flat prior on parameters would give a posterior $\pi(\theta|y) \propto L(\theta)$ and an approximate posterior $\tilde{\pi}(\theta|y) \propto \tilde{L}(\theta)$.

(Approximate) posterior distribution

Putting a flat prior on parameters would give a posterior $\pi(\theta|y) \propto L(\theta)$ and an approximate posterior $\tilde{\pi}(\theta|y) \propto \tilde{L}(\theta)$.



Divergence between approximate and exact posterior

We can find

$$KL(\tilde{\pi}(.|y), \pi(.|y)) = \int \pi(\theta|y) \log \frac{\pi(\theta|y)}{\tilde{\pi}(\theta|y)} d\theta$$

KL $= 0.18$

Divergence between approximate and exact posterior

We can find

$$KL(\tilde{\pi}(.|y), \pi(.|y)) = \int \pi(\theta|y) \log \frac{\pi(\theta|y)}{\tilde{\pi}(\theta|y)} d\theta$$

KL $=$ 0.18 (0.0085 for `two_level_10`)

Divergence between approximate and exact posterior

We can find

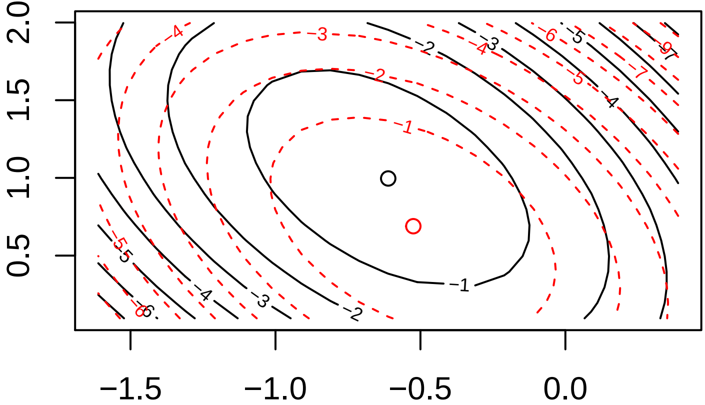
$$KL(\tilde{\pi}(.|y), \pi(.|y)) = \int \pi(\theta|y) \log \frac{\pi(\theta|y)}{\tilde{\pi}(\theta|y)} d\theta$$

KL $= 0.18$ (0.0085 for `two_level_10`)

Difficult to compute KL, even if we can compute $L(\theta)$ easily

Using a Normal approximation to the posterior

Approximate $\pi(.|y)$ with $N_p(\hat{\theta}, \hat{J}^{-1})$ and $\tilde{\pi}(.|y)$ with $N_p(\tilde{\theta}, \tilde{J}^{-1})$



Approximate divergence

We get the approximate KL in closed form, as

$$\frac{1}{2}\left[\log\frac{|\hat{J}|}{|\tilde{J}|} + \mathrm{tr}(\tilde{J}\hat{J}^{-1}) + (\tilde{\theta}-\hat{\theta})^T\tilde{J}(\tilde{\theta}-\hat{\theta})\right].$$

Approximate KL $= 0.21$ (0.0089 for `two_level_10`)

Approximate divergence

We get the approximate KL in closed form, as

$$\frac{1}{2}\left[\log\frac{|\hat{J}|}{|\tilde{J}|} + \mathrm{tr}(\tilde{J}\hat{J}^{-1}) + (\tilde{\theta}-\hat{\theta})^T\tilde{J}(\tilde{\theta}-\hat{\theta})\right].$$

Approximate KL $= 0.21$ (0.0089 for `two_level_10`)

Problem: need $\hat{\theta}$ and $\hat{J}$.

Using a one-step estimator

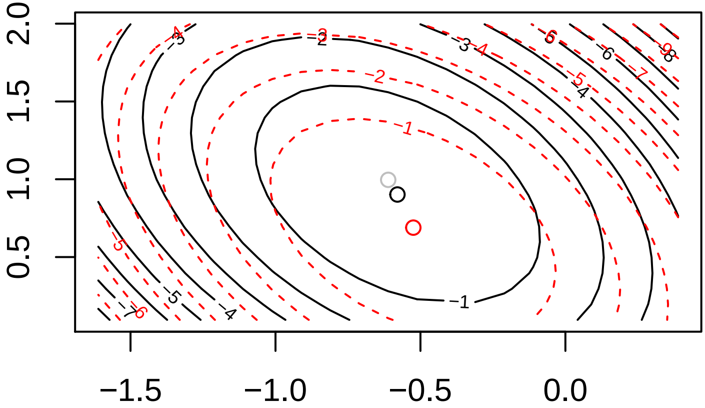
To avoid using $\hat{\theta}$, want to quickly find an estimate which moves towards $\hat{\theta}$ from $\tilde{\theta}$.

Let

$$\hat{\theta}_{(1)} = \tilde{\theta} + \tilde{J}^{-1} \nabla_{\theta} \ell(\tilde{\theta})$$

and approximate $\pi(.|y)$ with $N_p(\hat{\theta}_{(1)}, \tilde{J}^{-1})$

Comparing normal approximation



Approximate divergence

The approximate KL reduces to

$$\frac{1}{2}\left[(\tilde{\theta}-\hat{\theta}_{(1)})^{T}\tilde{J}(\tilde{\theta}-\hat{\theta})\right]$$

Approximate KL $= 0.085$ (0.0087 for `two_level_10`)

Need only first derivative of log-likelihood at $\tilde{\theta}$.

Second-order Laplace approximation

All of the above assumes that a very accurate approximation to the likelihood is available. But that won't always be the case.

Second-order Laplace approximation

All of the above assumes that a very accurate approximation to the likelihood is available. But that won't always be the case.

Instead, in `glmmsr` I have used a second-order Laplace approximation as a proxy for the exact likelihood.

```
c(fit_Laplace$laplace_divergence, fit_Laplace_10$laplace_divergence)
```

```
## [1] 0.21320343 0.01798935
```

The warnings are triggered if this divergence is greater than a threshold, by default 0.1.

Second-order Laplace approximation

The second order Laplace approximation to the log-likelihood is

$$\tilde{\ell}_{(2)}(\theta) = \tilde{\ell}(\theta) + \frac{1}{8}\hat{\kappa}_4 - \frac{1}{24}(2\hat{\kappa}_{23}^2 + 3\hat{\kappa}_{13}^2),$$

where

$$\hat{\kappa}_4 = \sum_{i,j,k,l} \hat{g}_{ijkl}\hat{g}^{ij}\hat{g}^{kl},$$

$$\hat{\kappa}_{13}^2 = \sum_{i,j,k,l,m,n} \hat{g}_{ijk}\hat{g}_{lmn}\hat{g}^{ij}\hat{g}^{kl}\hat{g}^{mn},$$

and

$$\hat{\kappa}_{23}^2 = \sum_{i,j,k,l,m,n} \hat{g}_{ijk}\hat{g}_{lmn}\hat{g}^{il}\hat{g}^{km}\hat{g}^{kn}.$$

Second-order Laplace approximation

Cost of naive computation: $O(m) + O(n^6)$ if there are m observations and n random effects.

Second-order Laplace approximation

Cost of naive computation: $O(m) + O(n^6)$ if there are m observations and n random effects.

There are various ways to simplify the computation.

Zipunnikov, V., & Booth, J. G. (2011). Closed form GLM cumulants and GLMM fitting with a SQUAR-EM-LA 2 algorithm

Changes cost to $O(m^2) + O(n^2)$. Currently used in `glmmsr`.

Next steps / Link with TMB

- TMB also allows inference with Laplace approximations: can cases where this might be a problem be detected within TMB?

Next steps / Link with TMB

- TMB also allows inference with Laplace approximations: can cases where this might be a problem be detected within TMB?
- Is the suggested approximation to KL divergence a sufficiently good way to check quality of Laplace inference? Is my choice of threshold suitable as a default?

Next steps / Link with TMB

- TMB also allows inference with Laplace approximations: can cases where this might be a problem be detected within TMB?
- Is the suggested approximation to KL divergence a sufficiently good way to check quality of Laplace inference? Is my choice of threshold suitable as a default?
- Can the second-order Laplace approximation be computed more efficiently? Can sparsity be exploited?

Next steps / Link with TMB

- TMB also allows inference with Laplace approximations: can cases where this might be a problem be detected within TMB?
- Is the suggested approximation to KL divergence a sufficiently good way to check quality of Laplace inference? Is my choice of threshold suitable as a default?
- Can the second-order Laplace approximation be computed more efficiently? Can sparsity be exploited?
- Automatic differentiation for second-order Laplace approximation?

Next steps / Link with TMB

- TMB also allows inference with Laplace approximations: can cases where this might be a problem be detected within TMB?
- Is the suggested approximation to KL divergence a sufficiently good way to check quality of Laplace inference? Is my choice of threshold suitable as a default?
- Can the second-order Laplace approximation be computed more efficiently? Can sparsity be exploited?
- Automatic differentiation for second-order Laplace approximation?
- In cases where we get a warning, how should we select an alternative approximation method?