Fitting GLMMs with `glmmsr`

Helen Ogden

Introduction

Generalized linear mixed models (GLMMs) are an important and widely-used model class. In R, it is possible to fit these models with the `lme4` package (Bates et al. 2015), but there are some limitations. First, except in very simple cases, `lme4` uses a Laplace approximation to the likelihood for inference, which may be of poor quality in some cases. Second, it is difficult to fit some GLMMs, such as pairwise comparison models, with `lme4`. The `glmmsr` package offers progress on both of these problems.

Approximations to the likelihood

In order to fit a GLMM with `glmmsr`, a user must first choose which method to use to approximate the likelihood. In addition to the Laplace and adaptive Gaussian quadrature approximations, which are borrowed from `lme4`, the likelihood may be approximated by the sequential reduction approximation (Ogden 2015), or an importance sampling approximation. These methods provide an accurate approximation to the likelihood in some situations where it is not possible to use adaptive Gaussian quadrature.

Example: a two-level model

Suppose that our data are clustered, so that we observe y_{ij} for $i = 1, \ldots, m$ and $j = 1, \ldots, m_i$, where y_{ij} is the jth observation from the ith cluster. We model $Y_{ij} \sim \text{Bernoulli}(p_{ij})$ where

$$\log\left(\frac{p_{ij}}{1-p_{ij}}\right) = \alpha + \beta x_{ij} + \sigma u_i,$$

and $u_i \sim N(0, 1)$. The artificial data `two_level`, available in `glmmsr`, is simulated from a binary two-level model with $m_i = 2$ observations per cluster, and with $m = 50$ clusters in total.

We want to infer the parameters $\theta = (\alpha, \beta, \sigma)$ of the model. The likelihood may be written as

$$L(\theta) = \int_{\mathbb{R}^m} g(\mathbf{u}; \theta) d\mathbf{u},$$

where

$$g(u; \theta) = \prod_{i=1}^{m} \prod_{j=1}^{m_i} Pr(Y_{ij} = y_{ij} | u_i, \theta) \phi(u_i).$$

For this model, and for other GLMMs, the likelihood function is an integral over the latent variables. It is not possible to evaluate this integral analytically, except in the case of a linear mixed model. In `glmmsr`, the interface for fitting this model is

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

where the choice of `method` determines the approximation to the likelihood. There is no default for `method`. This is a deliberate choice, to highlight to the user that some approximation must be used, and that the choice of this approximation might affect the resulting inference.

The Laplace approximation (`method = "Laplace"`)

The Laplace approximation works by approximating the integrand $g(\mathbf{u}; \theta)$ by a function proportional to a Gaussian density.

```
library(glmmsr)
mod_Laplace <- glmm(response ~ covariate + (1 | cluster), data = two_level,
                    family = binomial, method = "Laplace")
```

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

```
mod_Laplace
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Laplace approximation (lme4)
##
## Family: binomial ( logit )
## Formula: response ~ covariate + (1 | cluster)
##
## Random effects:
##   Groups  Name        Estimate
## 1 cluster (Intercept) 0.7484
## Number of obs: 100, groups: cluster, 50;
##
## Fixed effects:
## (Intercept)   covariate
##      0.6525     -1.1583
```

This gives some estimates of the parameters, but we do not know if these are close to the maximum likelihood estimates. Typically the structure of the data gives some hints about the quality of the Laplace approximation: if the data are 'sparse', in that there is only a small number of observations per random effect, the Laplace approximation may be a poor approximation to the likelihood. This is certainly the case here, since we only have two observations per cluster, and we would like to try using a more accurate approximation to the likelihood.

Adaptive Gaussian quadrature (`method = "AGQ"`)

For a two-level model, where each observation is contained within a single cluster, the likelihood simplifies to a product of one-dimensional integrals:

$$L(\theta) = \prod_{i=1}^{m} \int \prod_{j=1}^{m_i} Pr(Y_{ij} = y_{ij} | u_i, \theta) \phi(u_i) du_i.$$

In this case, we can use adaptive Gaussian quadrature with `nAGQ` points to approximate each of the one-dimensional integrals.

```
glmm(response ~ covariate + (1 | cluster), data = two_level,
     family = binomial, method = "AGQ", control = list(nAGQ = 16),
     prev_fit = mod_Laplace)
```

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

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Adaptive Gaussian Quadrature with 16 points (lme4)
##
## Family: binomial ( logit )
## Formula: response ~ covariate + (1 | cluster)
##
## Random effects:
##   Groups  Name        Estimate
## 1 cluster (Intercept) 1.041
## Number of obs: 100, groups: cluster, 50;
##
## Fixed effects:
## (Intercept)   covariate
##      0.7168     -1.2734
```

The estimate of the random-effect standard deviation is significantly larger than that obtained using the Laplace approximation to the likelihood.

Unfortunately, we can only use this method for a simple two-level model. Next, we consider a situation where we cannot use this method.

Example: three-level model

Now suppose that each of the clusters is itself contained within a larger group, so that we observe y_{ijk} for $i = 1, \ldots, m$, $j = 1, \ldots, m_i$ and $k = 1, \ldots, m_{ij}$, where y_{ijk} represents the kth observation in the jth cluster of the ith group. We have $Y_{ijk} \sim \text{Bernoulli}(p_{ijk})$ where

$$\log\left(\frac{p_{ijk}}{1-p_{ijk}}\right) = \alpha + \beta x_{ijk} + b_i + c_j,$$

$b_i \sim N(0, \sigma_b^2)$ and $c_j \sim N(0, \sigma_c^2)$.

The artificial data `three_level`, available in `glmmsr`, is simulated from a binary two-level model with $m_{ij} = 2$ observations per cluster, $m_i = 2$ clusters per group, and $m = 50$ clusters in total.

We may fit the model with the Laplace approximation to the likelihood:

```
mod_3_Laplace <- glmm(response ~ covariate + (1 | cluster) + (1 | group),
                      data = three_level, family = binomial, method = "Laplace")
```

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

```
mod_3_Laplace
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Laplace approximation (lme4)
##
## Family: binomial ( logit )
## Formula: response ~ covariate + (1 | cluster) + (1 | group)
##
## Random effects:
##   Groups  Name        Estimate
## 1 cluster (Intercept) 0.3576
```

```
## 2 group   (Intercept) 0.4257
## Number of obs: 200, groups: cluster, 100; group, 50;
##
## Fixed effects:
## (Intercept)   covariate
##     -0.1909      0.1199
```

The structure is sparse – there is little information provided by the data about the value of each random effect – so we might again question whether the Laplace approximation is of sufficiently high quality.

We could try to increase the accuracy of the approximation by using adaptive Gaussian quadrature to approximate the integral.

```
glmm(response ~ covariate + (1 | cluster) + (1 | group), data = three_level,
     family = binomial, method = "AGQ", control = list(nAGQ = 16))
```

```
## Error in lme4::updateGlmerDevfun(devfun_lme4, modfr$reTrms, nAGQ = nAGQ_lme4): nAGQ > 1 is only avai
```

We get an error message, because the likelihood does not reduce to a product of one-dimensional integrals in this case, as it does for a two-level model.

The sequential reduction approximation (`method = "SR"`)

The sequential reduction approximation to the likelihood is described in Ogden (2015).

The approximation is controlled by a parameter `nSL`, the level of sparse grid storage. `nSL` is a non-negative integer, where `nSL = 0` corresponds to the Laplace approximation, and increasing `nSL` gives a more accurate approximation to the likelihood. In the special case of a two-level model, sequential reduction is equivalent to adaptive Gaussian quadrature with $(2^{n_{SL}+1} - 1)$ quadrature points. Typically `nSL = 3` is large enough to give inference fairly close to the exact likelihood inference, although this varies according to the structure of the model.

In the three-level example, `glmm` may be used to fit the model using the sequential reduction approximation, with 3 sparse grid levels.

```
mod_3_SR <- glmm(response ~ covariate + (1 | cluster) + (1 | group),
                 data = three_level, family = binomial, method = "SR",
                 control = list(nSL = 3), prev_fit = mod_3_Laplace)
```

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

```
mod_3_SR
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Sequential reduction at level 3
##
## Family: binomial ( logit )
## Formula: response ~ covariate + (1 | cluster) + (1 | group)
##
## Random effects:
##   Groups  Name        Estimate
## 1 cluster (Intercept) 0.6463
```

```
## 2 group   (Intercept) 0.4502
## Number of obs: 200, groups: cluster, 100; group, 50;
##
## Fixed effects:
## (Intercept)   covariate
##     -0.2077      0.1389
```

The estimates of the random effects standard deviations are larger than the corresponding estimates from the Laplace approximation.

To check the quality of the approximation, we increase the level of sparse grid storage

```
mod_3_SR_4 <- glmm(response ~ covariate + (1 | cluster) + (1 | group),
                   data = three_level, family = binomial, method = "SR",
                   control = list(nSL = 4), prev_fit = mod_3_SR)
```

```
## Approximating the likelihood at each point takes 0.212 seconds.
## Fitting the model.............. done.
```

We see that the parameter estimates are very close to those obtained with the lower level of sparse grid storage, and conclude that these estimates are probably close to the exact maximum likelihood estimates.

Example: `salamander` data

McCullagh and Nelder (1989) discuss an experiment designed to study whether salamanders from two different populations would breed with one another. This well-known salamander mating data set is available in the `hglm.data` package.

```
data(salamander, package = "hglm.data")
```

Here, we fit the model used by Booth and Hobert (1999), and many other authors. Writing Y_{ij} for an indicator of whether the female salamander i mates with male salamander j, we have $Y_{ij} \sim \text{Bernoulli}(p_{ij})$, where

$$\log\left(\frac{p_{ij}}{1-p_{ij}}\right) = \beta^T \mathbf{x}_{ij} + b_i + c_j,$$

and $b_i \sim N(0, \sigma_f^2)$, $c_j \sim N(0, \sigma_m^2)$. Here $\mathbf{x}_{ij}$ tells us which population each of the pair of salamanders belongs to, given by the variable `Cross` in the `salamander` data. The latent variables b_i and c_j represent the differences in propensity to mate between individual animals, not explained by the population effects.

We can fit the model using the Laplace approximation:

```
mod_sal_Laplace <- glmm(Mate ~ 0 + Cross + (1 | Male) + (1 | Female),
                        family = binomial, data = salamander, method = "Laplace")
```

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

```
mod_sal_Laplace
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Laplace approximation (lme4)
##
```

```
## Family: binomial ( logit )
## Formula: Mate ~ 0 + Cross + (1 | Male) + (1 | Female)
##
## Random effects:
##   Groups Name        Estimate
## 1 Male   (Intercept) 1.020
## 2 Female (Intercept) 1.084
## Number of obs: 360, groups: Male, 60; Female, 60;
##
## Fixed effects:
## CrossRR CrossRW CrossWR CrossWW
##  1.0082  0.3062 -1.8960  0.9904
```

We can fit the model using the sequential reduction approximation with 2 sparse levels, although this is quite time-consuming:

```
mod_sal_SR_2 <- glmm(Mate ~ 0 + Cross + (1 | Male) + (1 | Female),
                     family = binomial, data = salamander, method = "SR",
                     control = list(nSL = 2), prev_fit = mod_sal_Laplace)
```

If we try to use 3 sparse levels, we get an error:

```
mod_sal_SR_3 <- glmm(Mate ~ 0 + Cross + (1 | Male) + (1 | Female),
                     family = binomial, data = salamander, method = "SR",
                     control = list(nSL = 3))
```

```
## Error: The sequential reduction approximation with 3 sparse levels
##  is too difficult to compute in this case.
##  Consider reducing nSL, or using a different approximation method.
```

The sequential reduction approximation is too expensive to compute in this case.

Importance sampling (`method = "IS"`)

If the sequential reduction approximation is too difficult to compute, we can use an importance sampling approximation to the likelihood.

This takes `nIS` samples $\mathbf{u}^{(j)}$ from the Gaussian approximation $N(\mu_\theta, \Sigma_\theta)$ used in the Laplace approximation, and approximates $L(\theta)$ with

$$\hat{L}_{n_{\text{IS}}}(\theta) = \sum_{j=1}^{n_{\text{IS}}} \frac{g(\mathbf{u}^{(j)}; \theta)}{\phi(\mathbf{u}^{(j)}; \mu_\theta, \Sigma_\theta)}.$$

To ensure that the approximation to the likelihood surface is smooth, the samples used for each value of θ are constructed by transforming a common set of standard normal samples.

Since this is a stochastic approximation, we will obtain different fits with different random seeds. We fit our model for the `salamander` data using importance sampling with 1000 samples:

```
set.seed(1)
mod_sal_IS_1000 <- glmm(Mate ~ 0 + Cross + (1 | Male) + (1 | Female),
                        family = binomial, data = salamander,
                        method = "IS", control = list(nIS = 1000),
                        prev_fit = mod_sal_Laplace)
```

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

```
mod_sal_IS_1000
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Importance sampling with 1000 samples
##
## Family: binomial ( logit )
## Formula: Mate ~ 0 + Cross + (1 | Male) + (1 | Female)
##
## Random effects:
##   Groups Name        Estimate
## 1 Male   (Intercept) 1.111
## 2 Female (Intercept) 1.166
## Number of obs: 360, groups: Male, 60; Female, 60;
##
## Fixed effects:
## CrossRR CrossRW CrossWR CrossWW
##  1.0146  0.3175 -1.9323  0.9962
```

We observe a small increase in the estimate of the random-effects standard deviation when we this importance sampling approximation to the likelihood. We could increase `nIS` even further, to check that the inference is close to exact likelihood inference.

The subformula interface

The interface of `glmm` allows fitting of some more complex models which are not easy to fit with `lme4`.

A typical call using this extended interface looks like `glmm(formula, subformula, data, family, method)` where

1. `formula` may contain one or more terms surrounded with `Sub(.)`. We call the expression contained within `Sub(.)` a **substitution expression**. This is a mathematical expression dictating how the response depends on a **substituted variable**: a dummy variable not contained in `data`.
2. `subformula` contains a **subformula** for each substituted variable, which describes how the substituted variable depends on covariates.

Next, we consider an example of the type of model we might want to fit using this interface.

Pairwise competition models

Suppose that we observe the outcome of a set of matches played between pairs of players, and that we also observe some covariates x_i for each player i. We suppose that each player i has an 'ability' λ_i, and that

$$Pr(i \text{ beats } j|\lambda_i, \lambda_j) = g(\lambda_i - \lambda_j),$$

where $g(.)$ is an inverse link function. If we are interested in how the ability depends on the covariates, we might model

$$\lambda_i = \beta x_i + b_i,$$

where $b_i \sim N(0, \sigma^2)$, and β and σ are unknown parameters.

This is a structured pairwise competition model. The `BradleyTerry2` package (Turner and Firth 2012) provides a good interface to fit these models, but it uses Penalized Quasi Likelihood (PQL) for inference if there are random effects in the model. PQL is often a poor approximation to the true likelihood.

We wrote down the structured pairwise competition model using a two-step approach: first we described how the response depends on the unknown abilities, then we wrote down how the abilities depend on covariates. This type of model can be written quite naturally using the subformula interface. We have a formula `response ~ 0 + Sub(ability[player1] - ability[player2])`, where `ability` is a substitution variable. We write down a corresponding subformula `ability[i] ~ 0 + x[i] + (1 | i)`. Here `player1` and `player2` are factors with common levels, and `x` is a vector of player-specific covariates. The common levels of `player1` and `player2` give the full vector of players involved in the tournament, and the covariates should ordered so that the ith component of `x` gives covariates for the ith player in this vector. The symbol used index the players, in this case `i`, is arbitrary.

Example: `flatlizards` data

Whiting et al. (2006) study contests between male flat lizards. The data are available in `BradleyTerry2`: see `?flatlizards` for more details. The aim of the study was to determine which covariates affected the fighting 'ability' of a lizard. We consider a simplified example, using only the covariate `SVL`, the snout vent length of the lizard. Unlike many of the other predictors, `SVL` has no missing values.

The model may be analysed using `BradleyTerry2`.

```
library(BradleyTerry2)
result <- rep(1, nrow(flatlizards$contests))
lizards_mod_BTm <- BTm(result, winner, loser, ~ SVL[..] + (1|..),
                       family = binomial(link = "probit"), data = flatlizards)
summary(lizards_mod_BTm)
```

```
##
## Call:
##
## BTm(outcome = result, player1 = winner, player2 = loser, formula = ~SVL[..] +
##     (1 | ..), family = binomial(link = "probit"), data = flatlizards)
##
## Fixed Effects:
##         Estimate Std. Error z value Pr(>|z|)
## SVL[..]  0.13184    0.06818   1.934   0.0532 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Random Effects:
##           Estimate Std. Error z value Pr(>|z|)
## Std. Dev.   0.6811     0.1531   4.449 8.62e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations: 8
```

We can fit the same model using `glmmsr`, first using the Laplace approximation to the likelihood.

```
flatlizards_glmmsr <- c(list(result = result,
                             winner = flatlizards$contests$winner,
                             loser = flatlizards$contests$loser),
                        flatlizards$predictors)
lizards_mod_Laplace <- glmm(result ~ 0 + Sub(ability[winner] - ability[loser]),
                            ability[liz] ~ 0 + SVL[liz] + (1 | liz),
                            data = flatlizards_glmmsr, family = binomial(link = "probit"),
                            method = "Laplace")
```

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

```
summary(lizards_mod_Laplace)
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Laplace approximation (lme4)
##
## Family: binomial ( probit )
## Formula: result ~ 0 + Sub(ability[winner] - ability[loser])
## Subformula: ability[liz] ~ 0 + SVL[liz] + (1 | liz)
##
## Random effects:
##  Groups Name        Estimate Std.Error
##  liz    (Intercept) 1.201    0.09716
## Number of obs: 100, groups: liz, 77;
##
## Fixed effects:
##          Estimate Std. Error z value Pr(>|z|)
## SVL[liz]   0.2467    0.09435   2.615 0.008933
```

We can try using a more accurate likelihood approximation, such as sequential reduction with `nSL = 2`.

```
lizards_mod_SR_2 <- glmm(result ~ 0 + Sub(ability[winner] - ability[loser]),
                         ability[liz] ~ 0 + SVL[liz] + (1 | liz),
                         data = flatlizards_glmmsr, family = binomial(link = "probit"),
                         method = "SR", control = list(nSL = 2),
                         prev_fit = lizards_mod_Laplace)
```

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

```
summary(lizards_mod_SR_2)
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Sequential reduction at level 2
##
## Family: binomial ( probit )
## Formula: result ~ 0 + Sub(ability[winner] - ability[loser])
## Subformula: ability[liz] ~ 0 + SVL[liz] + (1 | liz)
##
## Random effects:
##  Groups Name        Estimate Std.Error
```

```
##  liz    (Intercept) 1.835    0.5273
## Number of obs: 100, groups: liz, 77;
##
## Fixed effects:
##          Estimate Std. Error z value Pr(>|z|)
## SVL[liz]   0.2825     0.1719   1.643   0.1003
```

We can increase `nSL` further.

```
lizards_mod_SR_3 <- glmm(result ~ 0 + Sub(ability[winner] - ability[loser]),
                         ability[liz] ~ 0 + SVL[liz] + (1 | liz),
                         data = flatlizards_glmmsr, family = binomial(link = "probit"),
                         method = "SR", control = list(nSL = 3),
                         prev_fit = lizards_mod_SR_2)
```

```
## Approximating the likelihood at each point takes 0.508 seconds.
## Fitting the model..................................... done.
```

```
summary(lizards_mod_SR_3)
```

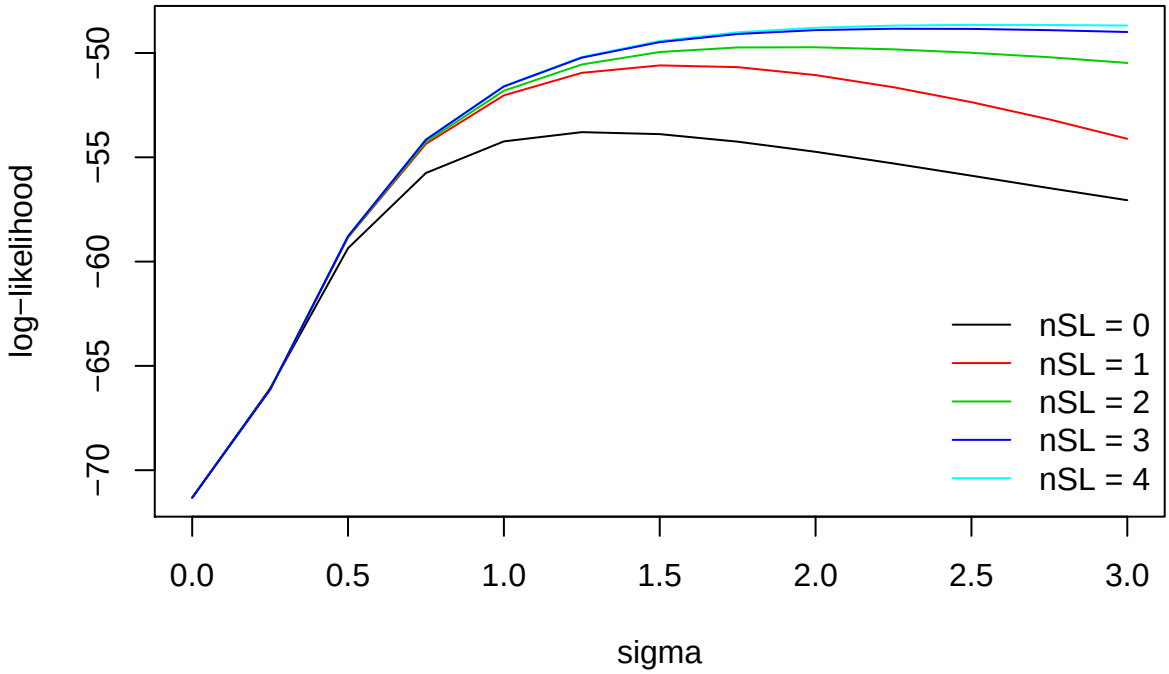
```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Sequential reduction at level 3
##
## Family: binomial ( probit )
## Formula: result ~ 0 + Sub(ability[winner] - ability[loser])
## Subformula: ability[liz] ~ 0 + SVL[liz] + (1 | liz)
##
## Random effects:
##  Groups Name        Estimate Std.Error
##  liz    (Intercept) 2.601    0.655
## Number of obs: 100, groups: liz, 77;
##
## Fixed effects:
##          Estimate Std. Error z value Pr(>|z|)
## SVL[liz]   0.3837     0.1993   1.925  0.05425
```

The inference we get using different approximations to the likelihood is very different. As we increase `nSL`, the estimate of the random-effect standard deviation σ increases.

To investigate why this happens, we examine a cut across the various approximate log-likelihood surfaces, at $\beta = 0.3$, for various values of σ.

```
modfr_lizards <-  find_modfr_glmm(result ~ 0 + Sub(ability[winner] - ability[loser]),
                                  ability[liz] ~ 0 + SVL[liz] + (1 | liz),
                                  data = flatlizards_glmmsr,
                                  family = binomial(link = "probit"))
theta_poss <- cbind(seq(0, 3, by = 0.25), 0.3)
l_SR_theta_poss <- list()
for(i in 0:4) {
  lfun_SR_i <- find_lfun_glmm(modfr_lizards, method = "SR", control = list(nSL = i))
  l_SR_theta_poss[[i + 1]] <- apply(theta_poss, 1, lfun_SR_i)
}
```

```
plot(theta_poss[,1], l_SR_theta_poss[[5]], type = "l", col = 5,
     xlab = "sigma", ylab = "log-likelihood")
for(i in 1:4) {
  lines(theta_poss[,1], l_SR_theta_poss[[i]], col = i)
}
legend("bottomright", legend = paste("nSL =", 0:4), col = 1:5, lty = 1, bty = "n")
```



The log-likelihood $\ell(\sigma, \beta = 0.3)$ is an increasing function of σ, but the less accurate approximations to the likelihood substantially underestimate the likelihood for large values of σ, and so mask this difficulty. We will not discuss here how to deal with this problem, but note that without access to an accurate approximation to the likelihood, this problem would have gone unnoticed.

Example: `salamander` data revisited

Recall that we assume that female i and male j mated with probability p_{ij}, where

$$\log\left(\frac{p_{ij}}{1-p_{ij}}\right) = \beta^T \mathbf{x}_{ij} + b_i + c_j.$$

Previously, we assumed that $b_i \sim N(0, \sigma_f^2)$ and $c_j \sim N(0, \sigma_m^2)$ are samples from different distributions. We might also be interested in a simpler model, where b_i and c_j are both samples from a common $N(0, \sigma^2)$ distribution.

We may fit this model using the subformula interface of `glmmsr`. To do this, we will use a substituted variable `propen`, representing the latent propensity of each salamander to mate. First, we must construct identifiers for the female and the male salamander involved in each match. These identifiers should be factors with common levels, where the levels are the identifiers of all the salamanders involved in the experiment:

```
female_id <- paste("F", salamander$Female, sep = "")
male_id <- paste("M", salamander$Male, sep = "")
ids <- unique(c(female_id, male_id))
```

```
salamander$female_id <- factor(female_id, levels = ids)
salamander$male_id <- factor(male_id, levels = ids)
```

We may then fit the model, using a Laplace approximation to the likelihood:

```
mod_sal_equal <- glmm(Mate ~ 0 + Cross + Sub(propen[female_id] + propen[male_id]),
                      propen[sal] ~ 0 + (1 | sal), family = binomial,
                      data = salamander, method = "Laplace")
```

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

```
mod_sal_equal
```

```
## Generalized linear mixed model fit by maximum likelihood [glmmFit]
## Likelihood approximation: Laplace approximation (lme4)
##
## Family: binomial ( logit )
## Formula: Mate ~ 0 + Cross + Sub(propen[female_id] + propen[male_id])
## Subformula: propen[sal] ~ 0 + (1 | sal)
##
## Random effects:
##   Groups Name        Estimate
## 1 sal    (Intercept) 1.052
## Number of obs: 360, groups: sal, 120;
##
## Fixed effects:
## CrossRR CrossRW CrossWR CrossWW
##  1.0098  0.3061 -1.8953  0.9882
```

Conclusions

The `glmmsr` package provides the user with a choice of methods for approximating the likelihood, including the sequential reduction approximation (Ogden 2015), which gives an accurate approximation to the likelihood in many cases for which the Laplace approximation is unreliable. Some cases remain in which the sequential reduction approximation is too expensive to compute, despite the poor quality of the Laplace approximation. An importance sampling approximation may sometimes be used to obtain accurate inference in such cases, although if the Laplace approximation is a poor approximation, the importance sampling approximation might take a very long time to converge. This motivates the need for other methods for approximating the likelihood, and we hope to add such methods in future versions of `glmmsr`.

References

Bates, Douglas, Martin Maechler, Benjamin M. Bolker, and Steven Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." http://arxiv.org/abs/1406.5823.

Booth, James G, and James P Hobert. 1999. "Maximizing Generalized Linear Mixed Model Likelihoods with an Automated Monte Carlo EM Algorithm." *Journal of the Royal Statistical Society: Series B (Statistical*

Methodology) 61 (1). Wiley Online Library: 265–85.

McCullagh, Peter, and John A Nelder. 1989. *Generalized Linear Models*. 2nd ed. Vol. 37. Monographs on Statistics and Applied Probability. CRC press.

Ogden, Helen. 2015. "A sequential reduction method for inference in generalized linear mixed models." *Electronic Journal of Statistics* 9: 135–52. doi:10.1214/15-EJS991.

Turner, Heather, and David Firth. 2012. "Bradley-Terry Models in R: The BradleyTerry2 Package." *Journal of Statistical Software* 48 (9): 1–21. http://www.jstatsoft.org/v48/i09/.

Whiting, Martin J., Devi M. Stuart-Fox, David O'Connor, David Firth, Nigel C. Bennett, and Simon P. Blomberg. 2006. "Ultraviolet signals ultra-aggression in a lizard." *Animal Behaviour* 72 (2): 353–63. doi:10.1016/j.anbehav.2005.10.018.