

An introduction to the **spaMM** package for mixed models

François Rousset

August 30, 2023

The **spaMM** package fits mixed models. It was developed first to fit models with spatial correlations, which commonly occur in ecology. These correlations can be taken into account in generalized linear mixed models (GLMMs). However, there has been a dearth of validated software for making inferences under spatial GLMMs. This package has been first designed to fill this gap ([Rousset and Ferdy, 2014](#)). It now allows inferences under models with or without spatially-correlated random effects. It has been extended to handle multivariate-response models, and conditional response families beyond generalized linear models. Variation in residual variance (heteroscedasticity) can itself be represented by a mixed-effect model. Among spatial models, it can for example fit both random effects with the classical Matérn correlation function ([Matérn, 1960](#); [Stein, 1999](#)), and grid-based approximations of this model ([Lindgren et al., 2011](#)), for which INLA ([Lindgren and Rue, 2015](#)) is known.

spaMM thus provides a common interface for performing different analyses currently performed by different packages or difficult to perform by other means. It even provides a robust alternative function `spaMM_glm()` to the `glm()` function, suitable when the latter diverges or fails to find good starting values. Initial development drew inspiration from work by Lee and Nelder on h -likelihood, and it retains from that work several distinctive features, such as the ability to fit models with non-gaussian random effects, structured dispersion models, and implementation of several variants of Laplace and PQL approximations. However, later extensions mean that various alternative methods have been implemented.

As a first introduction, this document does not address all aspects of inference. A series of examples is presented in order to introduce the main functions, different types of models (spatial LMMs, GLMMs, and the wider

classes allowing non-gaussian random effects and more diverse response families), and some of the correlation structures available for random effects. Additional sections describe the approximations of likelihood used, the Conway-Maxwell-Poisson response family, multivariate-response models, some rarely needed controls of `spaMM` computations, and provide comparisons with alternative software.

The following concepts are assumed at least superficially known: generalized linear models (GLM), the basic syntax of the `glm` procedure in R, the concept of mixed-effect model, formal inference using likelihood-ratio tests, and the Greek alphabet (in particular, β for fixed-effect coefficients, ϕ for the variance of residual error, λ for the variance of random effects, but also μ and η to describe expectations of the response, and ρ and ν for correlation parameters).

Contents

1	Quick start for non-spatial models	3
2	An example of geostatistical analysis (spatial LMM)	4
2.1	Understanding and fitting the spatial model	4
2.2	Prediction	9
2.2.1	Point prediction	9
2.2.2	Prediction variance	9
3	General features of <code>spaMM</code>	11
3.1	Model formulation	11
3.2	Response families	13
3.2.1	Overview	13
3.2.2	The COMPoisson family	13
3.3	The main procedures in <code>spaMM</code>	14
3.3.1	Fitting functions	14
3.3.2	Post-fit functions	15
3.4	Multiple comparisons	17
3.5	Control of the fitting methods	19
3.5.1	Objective functions in ML, REML and PQL fits	19
3.5.2	Implications of REML for post-fit inferences	21
3.5.3	Inner iterations versus outer optimization	22
3.5.4	Numerical methods	23

4	Further examples	24
4.1	Spatial GLMMs	24
4.1.1	Basic syntax	24
4.1.2	A classic example with autoregressive random effects	24
4.2	Beta random effects and binomial logit-Beta model	26
4.2.1	Gamma GLMM, HGLM, and joint GLMs	27
4.3	Random-slope model	31
4.4	Multivariate response	33
4.5	Zero-altered (hurdle) models	35
4.5.1	Fitting	36
4.5.2	Point-prediction intervals	37
5	Comparison with alternative software and literature results	40
5.1	Procedures for geostatistical models (or contrived to such usage): <code>MASS::glmmPQL</code> , <code>lmer</code> , <code>geoRglm</code> , <code>glmmTMB</code>	41
5.2	Interpolated Markov random fields via <code>spaMM</code> and <code>INLA</code>	41
5.3	Gamma GLMM with non-canonical link	43
5.3.1	A comparison with published estimates	44
5.3.2	Further comparisons with <code>glmer</code> , and <code>glmmTMB</code>	45
5.3.3	PQL vs. <code>glmmPQL</code>	47
5.4	Negative binomial model	47
	Bibliography	49
	Appendices	52
A	Evaluation of the likelihood approximations	52
A.1	Conditional and joint log-likelihood	53
A.2	The gradient and Hessian matrix of conditional likelihood	54
A.3	Expected Hessian approximation for generalized linear models	55
A.4	Using observed versus expected Hessian	56
A.5	Hands-on computation of Laplace approximations	57
B	Multivariate analyses: the aster example	59
	Index	66

1 Quick start for non-spatial models

if you are used to `lme4`, then you can use `spaMM` almost identically by calling `fitme`. For example an LMM fit

```
data("Orthodont", package = "nlme")
library(lme4)
lfit <- lmer(distance ~ age+(age|Subject), data = Orthodont)
```

becomes

```
data("Orthodont", package = "nlme")
library(spaMM)
spfit <- fitme(distance ~ age+(age|Subject), data = Orthodont, method="REML")
```

The one argument to always think about is the `method` argument. as default fitting methods differ among packages, and even among fitting functions within packages. Gentle messages or warnings should attract one's attention on any other departure from deeply-established practice.

The story is the same for GLMMs, except that `glmer` has no REML method. Thus

```
data(salamander)
glfit <- glmer(cbind(Mate, 1-Mate) ~ TypeF+TypeM+TypeF*TypeM+(1|Female)
              +(1|Male), family=binomial(), data=salamander)
```

becomes

```
spfit <- fitme(cbind(Mate, 1-Mate) ~ TypeF+TypeM+TypeF*TypeM+(1|Female)
              +(1|Male), family=binomial(), data=salamander)
```

`spaMM` is routinely checked and used on large datasets and care is taken that it is fast. However, it is stubborn, meaning that if convergence is not reached after some moderate computation effort, `spaMM` tends to try harder, instead of terminating with a convergence warning. If you find a fit too long, then the argument `verbose=c(TRACE=TRUE)` may be useful to monitor progress of the computation. Further, `fitme` tries to select the fastest among several fitting strategies, but may not always select the best one. So, if a fit takes unduly long time, it is worth reading `help(convergence)`.

2 An example of geostatistical analysis (spatial LMM)

2.1 Understanding and fitting the spatial model

We fit data from a simple Gaussian model, according to which each response value y_i is assumed to be of the form

$$y_i = \text{fix}_i + b_i + e_i \quad (1)$$

where a fixed part fix_i represents effects of known predictor variables, and $b_i + e_i$ represent two Gaussian random terms with different correlation structures: e_i is a residual error with independent values for each observation, while b_i values can be correlated among different observations.

We first generate spatially correlated Gaussian-distributed data as follows

```
library(MASS)

rSample <- function(nb, rho, sigma2_u, resid, intercept, slope, pairs=TRUE) {
  ## sample pairs of adjacent locations
  if (pairs) {
    x <- rnorm(nb/2); x <- c(x, x+0.001)
    y <- rnorm(nb/2); y <- c(y, y+0.001)
  } else {x <- rnorm(nb); y <- rnorm(nb)}
  dist <- dist(cbind(x, y)) ## distance matrix between locations
  m <- exp(-rho*as.matrix(dist)) ## correlation matrix
  b <- mvrnorm(1, rep(0, nb), m*sigma2_u) ## correlated random effects
  pred <- sample(nb) ## some predictor variable
  obs <- intercept + slope*pred + b + rnorm(nb, 0, sqrt(resid)) ## response
  data.frame(obs=obs, x=x, y=y, pred=pred)
}

set.seed(123)
d1 <- rSample(nb=40, rho=3, sigma2_u=0.5, resid=0.5, intercept=-1, slope=0.1)
```

This has generated data in 2D (x, y) space with fixed effects $\text{fix}_i = -1 + 0.1 \text{pred}$ for some predictor variable `pred`, random effect variance 0.5, residual error variance 0.5, and b_i s whose correlations are exponentially decreasing with distance d as $\exp(-3d)$. Using standard notation for linear models, the fixed effects are written as $\mathbf{X}\boldsymbol{\beta}$ where $\boldsymbol{\beta} = (-1, 0.1)^\top$ and \mathbf{X} is the design matrix for fixed effects, here a two-column matrix whose first column is filled with 1 and the second with the variable `pred`. The random-effect variance will be denoted λ and the residual error variance will be denoted ϕ .

This model can be fitted as follows:

```
HLM <- fitme(obs~pred+Matern(1|x+y), data=d1, fixed=list(nu=0.5))
```

Here the `Matern(1|x+y)` formula term means that the Matérn correlation model is fitted to the data (for other correlation models, see Section 3.1). This is a very convenient model for spatial correlation, and includes the exponential $\exp(-\rho d)$ and the squared exponential $\exp(-\rho d^2)$ as special cases.

The Matérn model is described by two correlation parameters, the scale parameter ρ , and a “smoothness” parameter ν ($\nu = 0.5$ and $\nu \rightarrow \infty$ for exponential and squared exponential models, respectively). By declaring `fixed=list(nu=0.5)`, we have therefore fitted the model with exponential spatial correlation $\exp(-\rho d)$.

The ρ estimate together with the fixed ν are shown as `rho` and `nu` value in the output:

```
summary(HLM)

## formula: obs ~ pred + Matern(1 | x + y)
## ML: Estimation of corrPars, lambda and phi by ML.
##      Estimation of fixed effects by ML.
## Estimation of lambda and phi by 'outer' ML, maximizing logL.
## family: gaussian( link = identity )
## ----- Fixed effects (beta) -----
##              Estimate Cond. SE t-value
## (Intercept)   -1.631   0.34861  -4.678
## pred           0.105   0.01218   8.623
## ----- Random effects -----
## Family: gaussian( link = identity )
##              --- Correlation parameters:
##      1.nu      1.rho
## 0.500000 2.163672
##              --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##      x + y   : 0.3368
## # of obs: 40; # of groups: x + y, 40
## ----- Residual variance -----
## phi estimate was 0.64443
## ----- Likelihood values -----
##              logLik
## logL          (p_v(h)): -54.54889
```

The other parameters estimated (with standard errors) are the coefficients `beta` of the fixed effects, the variance `lambda` (here σ_u^2) of the random effects, and the residual variance `phi` (here σ_e^2). All estimates look reasonably close to the simulated values.

The output shows the marginal log-likelihood of the fitted model, also named as `p_v(h)` which more generally denotes a class of approximations developed for marginal log-likelihood. The restricted likelihood (not shown here) will likewise be denoted `p_beta,v`. For linear mixed models, `p_v` is exactly the log-likelihood, and `p_beta,v` is exactly the restricted log-likelihood.

The `confint` function can be used to obtain confidence intervals from a fitted model object, either by parametric bootstrap or for fixed effects, using the profile likelihood

```
confint(HLM,"pred") ## interval for the 'pred' coefficient

## lower pred upper pred
## 0.08007909 0.13134922
```

In general there is no reason to assume a given ν value, so we fit the full Matérn model by removing the `fixed` argument:

```
fitme(obs~pred+Matern(1|x+y),data=d1)

## formula: obs ~ pred + Matern(1 | x + y)
## ML: Estimation of corrPars, lambda and phi by ML.
##      Estimation of fixed effects by ML.
## Estimation of lambda and phi by 'outer' ML, maximizing logL.
## family: gaussian( link = identity )
## ----- Fixed effects (beta) -----
##              Estimate Cond. SE t-value
## (Intercept)  -1.636   0.34197  -4.784
## pred          0.105   0.01216   8.630
## ----- Random effects -----
## Family: gaussian( link = identity )
##              --- Correlation parameters:
##          1.nu      1.rho
## 0.8882648 3.2336336
##              --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    x + y   : 0.3288
## # of obs: 40; # of groups: x + y, 40
## ----- Residual variance -----
## phi estimate was 0.651654
## ----- Likelihood values -----
##                      logLik
## logL      (p_v(h)): -54.54011
```

The ν and ρ estimates now look very poor. Indeed, it is often easier to estimate $\sqrt{\nu}/\rho$ than each of these two parameters separately.

It may also be difficult to estimate the variances λ and ϕ separately, in particular if spatial correlations are weak, as noted above. Indeed, if b_i has no correlation structure, it is not separable from the residual error

term e_i unless there are repeated observations in the same spatial location, because if (using traditional notation)¹ $(b_i) \sim \mathcal{N}(0, \sigma_b^2 \mathbf{I})$ and $(e_i) \sim \mathcal{N}(0, \sigma_e^2 \mathbf{I})$, $(b_i + e_i) \sim \mathcal{N}[0, (\sigma_b^2 + \sigma_e^2) \mathbf{I}]$ is equally well explained by any σ_b^2 and σ_e^2 with given sum.

To illustrate another cause for poor estimation of variances, we draw a new sample

```
set.seed(123)
d2 <- rSample(nb=40, rho=3, sigma2_u=0.5, resid=0.5, intercept=-1,
              slope=0.1, pairs=FALSE)
```

In the previous simulation we had sampled pairs of adjacent locations in space and in the new one there is no such clustering. This tends to yield poorer estimates of λ and/or ϕ :

```
fitme(obs~pred+Matern(1|x+y), data=d2)

## formula: obs ~ pred + Matern(1 | x + y)
## ML: Estimation of corrPars, lambda and phi by ML.
##      Estimation of fixed effects by ML.
## Estimation of lambda and phi by 'outer' ML, maximizing logL.
## family: gaussian( link = identity )
## ----- Fixed effects (beta) -----
##              Estimate Cond. SE t-value
## (Intercept)  -1.1564  0.27995  -4.131
## pred          0.1075  0.01088   9.883
## ----- Random effects -----
## Family: gaussian( link = identity )
##              --- Correlation parameters:
##      1.nu      1.rho
## 16.66667 23.96106
##              --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##      x + y   : 0.4393
## # of obs: 40; # of groups: x + y, 40
## ----- Residual variance -----
## phi estimate was 0.290201
## ----- Likelihood values -----
##              logLik
## logL      (p_v(h)): -48.21512
```

¹ $X \sim \mathcal{N}(\mu, \sigma^2)$ means that X follows a gaussian distribution with given mean and variance.

2.2 Prediction

2.2.1 Point prediction

The `predict` function returns the predicted value of the response, say $\mathbf{x}\hat{\boldsymbol{\beta}} + \hat{b}_l$ for a new location l in space, where \mathbf{x} are given values of the predictor variables and \hat{b}_l the predicted value of the spatial random effect in that location. For a spatial Gaussian effect this is the expected value of the Gaussian deviate given the inferred random effects in the observed locations and the covariances of the spatial process between the new location and the observed locations.

In general, prediction requires as input new \mathbf{x} values, and new z values for each random effect (for block random effects, the grouping variable should thus be provided; and for spatial random effects, new spatial coordinates are required). Often one wishes to produce a nice map of predicted values without providing new \mathbf{x} in every possible location (e.g. Fig. 1). See the documentation of the `filled.mapMM` function for critical comments on how this is achieved.

2.2.2 Prediction variance

Prediction uncertainty can also be computed. In mixed model there is a realized latent value of the random effect b and the uncertainty in the linear prediction is often characterized by the *conditional* distribution, for each observation i , of the difference between the prediction, $\widehat{\text{fix}}_i + \hat{b}_i$, and $\text{fix}_i + b_i$ given the latent value b_i , rather than characterized by any marginal distribution over random draws of the random effect. Prediction uncertainty depends on the joint conditional variation of the random effects and of the fixed effects given the data. As an extreme example, if the fitted residual variance is null (so that the fitted model interpolates between observed points) and the data-generating residual variance is also null, $\text{fix}_i + b_i$ is known for each observation, which means that the conditional prediction variance in observed locations is null and that uncertainties in fixed effects and random effects must exactly balance each other.

The `predict(., variances)` argument, and several *ad hoc* functions provides control of output of the different quantities that can measure such uncertainty, and of their components. One can notably use `get_respVar()` to compute the *response variance* which is the (again, conditional) variance of the response over replicates, which is often referred to as prediction variance and includes residual variance. One can use `get_predVar()` or `get_cPredVar()` to compute the distinct *prediction variance* which is here

```

data(Loaloea)
lfit <- fitme( cbind(npos,ntot-npos) ~ elev1 + elev2 + elev3 +
  elev4 + maxNDVI1 + seNDVI + Matern(1|longitude+latitude),
  data=Loaloea, family=binomial() )

if (suppressPackageStartupMessages(require(maps,quietly=TRUE))) {
  ## 'maps' required for add.map=TRUE
  filled.mapMM(lfit, add.map=TRUE, plot.axes=quote({axis(1);axis(2)}),
    decorations=quote(points(pred[,coordinates],pch=15,cex=0.3)),
    plot.title=title(main="Inferred prevalence, North Cameroon",
    xlab="longitude", ylab="latitude"))
}

```

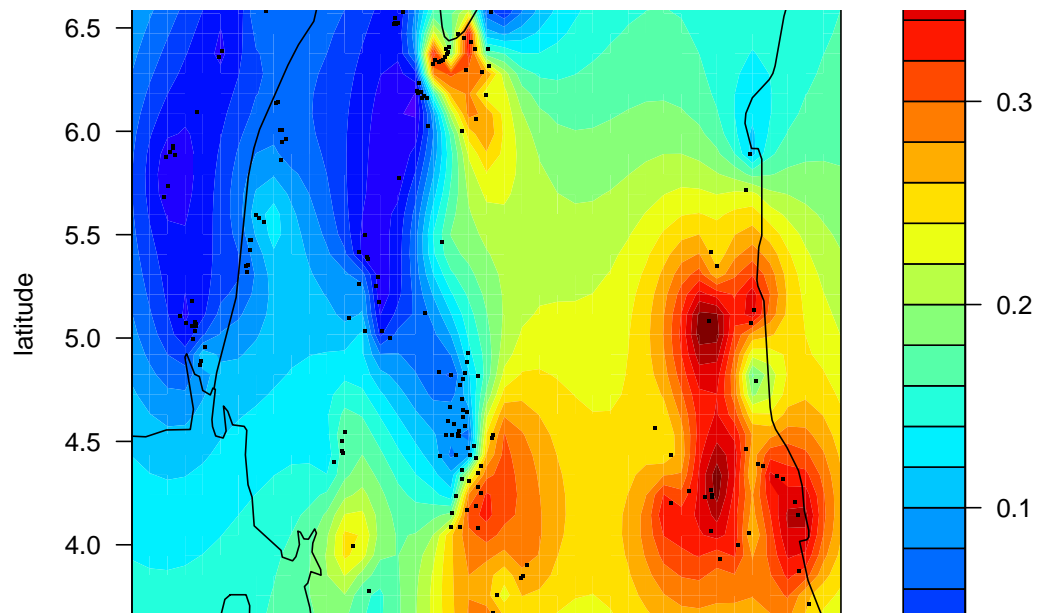


Figure 1: Plotting a map of predictions with `filled.mapMM`

the conditional variance of the linear predictor, excluding residual variance.² `get_predVar()` ignores a bias term discussed by Booth and Hobert (1998) while `get_cPredVar()` includes this bias, using a bootstrap procedure similar to but more general than the one discussed by these authors. The bootstrap procedure is computer-intensive, which may prevent its routine use for spatial prediction problems.

3 General features of spaMM

3.1 Model formulation

The `spaMM` output refers to the following formulation of all models, further illustrated in later examples. The expected response $\boldsymbol{\mu} = E(\mathbf{y}|\mathbf{b})$ given all realized random effects \mathbf{b} is written as the “linear predictor”

$$g(\boldsymbol{\mu}) = \boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{b} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{v} \text{ (plus any offset term)} \quad (2)$$

where g is the link function relating the linear predictor $\boldsymbol{\eta}$ to the expectation $\boldsymbol{\mu}$ of the response variable, and the structure of the random effects \mathbf{b} is described in terms of a vector \mathbf{v} with independent elements and of a “design matrix” \mathbf{Z} .³ \mathbf{v} can be further described as $\mathbf{v} = f(\mathbf{u})$ where f is another link function and the elements of \mathbf{u} are independent realizations of some reference distribution (e.g., gaussian). The fitting functions will provide estimates of fixed-effect parameters, and of the random-effect parameters classified as *dispersion* parameters (the variances of u_i and of the residual error e_i) and *correlation* parameters affecting the elements of \mathbf{Z} (ν and ρ in the previous examples).

In `spaMM`, implemented spatial random effects may follow the **Matern** or the **Cauchy** correlation models; or follow grid-based approximations of the Matern model (Lindgren et al., 2011, here denoted IMRFs for “Interpolated

²Here it should be emphasized that this departs from some established, but not universal, usage according to which “prediction variance” includes the residual variance, and this is what has here been called here the response variance. The idea that the prediction variance includes the residual variance is indeed emphasized in sources explaining the difference between confidence intervals and prediction intervals. Despite the excellence of some of these sources, such discussions may consider only fixed-effect models and thus teach a semantics that does not provide a name for a variance of prediction that does not include the residual variance. The “prediction intervals” in such semantics (our response intervals) should also be distinguished from “confidence intervals for predictions”, a term used in the literature for intervals for point predictions.

³Accordingly, the i th row of the expected response vector is denoted $g(\mu_i) = \eta_i = \mathbf{x}_i\boldsymbol{\beta} + b_i = \mathbf{x}_i\boldsymbol{\beta} + \mathbf{z}_i\mathbf{v}$. The i index will commonly be ignored.

Markov Random Fields”; see Section 5.2); or the variant introduced by Nychka et al. (2015), here denoted “multilevel IRMFs”, where several IMRFs are controlled by common hyper-parameters. Conditional autoregressive (CAR) models as described by an `adjacency` matrix can also be used in spatial analyses (see example in Section 4.1.2). Implemented time-series models include the classical $AR(p)$ and $ARMA(p, q)$ models. The $AR(1)$ model also has a distinct ad-hoc implementation as the `AR1` autoregressive model.

`spaMM` can fit random effects with any given correlation matrix (using the `corrMatrix` or `covStruct` arguments). This implies that it can fit any parametric correlation model once a wrapper function returning the likelihood for given parameters is fed to an optimizing function (such as the base R function `optim`, but `nloptr` or `minqa` optimizers should be considered for such purposes). This will generally be less computationally efficient than a “canned” implementation of a procedure for estimating the correlation parameters, but can still be very useful. However, any user-defined parametric correlation model can also be fitted efficiently if the (adventurous) user is willing to provide a `corrFamily constructor`, which is (roughly) a set of functions designed to interact efficiently with `spaMM`’s internal machinery. Several of the random effect models more recently introduced in `spaMM` are indeed implemented in this way, including one of the IMRF models (`MaternIMRFa`), the $AR(p)$ and $ARMA(p, q)$ models, and models for dyadic interactions (see the `ranGCA` and `diallel` documentation).

`spaMM` can fit random-coefficient terms (see Section 4.3) as well as *composite* random effects combining features of a `corrMatrix()` random effect and of a random-coefficient model: see Section 4.4 for a typical application in a quantitative-genetics model with a multivariate response. Composite random effects combining features of a parametric correlation model such as `Matern()` and of a random-coefficient model can also be fitted (The `Matern(z|x+y)` syntax, for a numeric variable `z`, has been left undefined in past versions of `spaMM` but is now interpreted as a composite random effect). In such composite random effects, the component correlation models are combined through an outer (Kronecker) product of component correlation matrices as detailed in `help("composite-ranef")`. In particular, `Matern(z|x+y)` means that for each spatial location i , a pair of random values $\mathbf{v}_i = (v_{i1}, v_{i2})$ is assumed. The v_{ij} ’s are correlated across i values for each j according to the Matérn correlation model, but the two realizations $\mathbf{v}_{.1}$ and $\mathbf{v}_{.2}$ are independent draws of the spatial process. A pair (b_{i1}, b_{i2}) is then computed from each pair \mathbf{v}_i as usual for random-slope models (section 4.3) so that the covariance among the elements of each new pair is as specified by the random-slope covariance matrix; and the b ’s enter into the linear predictor as later shown (eq. 7).

It is possible to fit some other “autocorrelated random-coefficient” mod-

els by a syntax analogous to that of other random-coefficient terms, providing direct control of the elements of the \mathbf{Z} matrix. For example, independent Matérn effects can be fitted for males and females by using the syntax `Matern(male|x+y) + Matern(female|x+y)`, where `male` and `female` are `TRUE|FALSE` factors (a sex effect should typically also be included in the fixed effects in such cases). A numeric variable `z` can also be considered, in which case the proper syntax is `Matern(0+z|x+y)`, which represents an autocorrelated random-slope term without random intercept (this syntax is equivalent to a direct specification of heteroscedasticity of the Matérn random effect).

These different random-effect specifications can all be combined in a model formula.

3.2 Response families

3.2.1 Overview

`spaMM` fits the following response families: the base GLM families `gaussian`, `Gamma`, `poisson`, `binomial`; the beta (`beta_resp`), beta-binomial (`betabin`) and two negative-binomial families; zero-truncated variants of the Poisson and negative-binomial families (see e.g. `help(Tpoisson)` for details); and the `COMPOisson` family, which handles an underdispersion parameter (further described below).

The negative binomial families include the non-GLM family `negbin1`, with linear mean-variance relationship, and the GLM family, `negbin` or `negbin2`, with quadratic mean-variance relationship. These families have a shape parameter, and their zero-truncated variants are handled, through a `trunc` argument (see also `help(Tnegbin)`).

`spaMM` also provides some facilities for the analysis of multinomial data. See `help(multinomial)` for more details.

3.2.2 The COMPOisson family

The Conway-Maxwell-Poisson family for count data is a generalization of the Poisson family that can describe over- and underdispersion, relative to Poisson response (e.g., [Shmueli et al., 2005](#)). The quasi-poisson method available in the `MASS` package has been used for such purposes, but it is not based on a probability model for count data. Overdispersion can also be represented by mixed models, but underdispersion in count data is less easy to represent, and the `COMPOisson` family is of particular interest in the latter

case. The distribution of a response y is

$$\Pr(y; \lambda, \nu_{\text{CMP}}) = \frac{\lambda^y}{(y!)^{\nu_{\text{CMP}}} Z(\lambda, \nu_{\text{CMP}})} \quad (3)$$

where $Z(\lambda, \nu_{\text{CMP}}) := \sum_{k=0}^{\infty} \lambda^k / (k!)^{\nu_{\text{CMP}}}$. The Poisson distribution is recovered for $\nu_{\text{CMP}} = 1$, in which case $Z = e^\lambda$, which also happens to be the mean μ of the distribution. However, for $\nu_{\text{CMP}} \neq 1$, the mean is not e^λ .

`spaMM` implements both the canonical link, which is complicated to evaluate,⁴ and some non-canonical ones (those implemented for the Poisson family, notably the log link). For fixed ν_{CMP} , the COMPoisson family is also of the form that can be fitted by `glm` (see `help("COMPoisson")` for examples using `glm`).

The main drawback of the COMPoisson is that the Z function has no expression in terms of standard “elementary” (efficiently implemented) functions, and involves an infinite summation that must be approximated by truncation. The number of terms required for accurate evaluation increases with decreasing ν_{CMP} . Further, the inverse function of Z (already needed for fitting by `glm`) has no explicit expression. Altogether, this implies that (depending on the quality of the approximations used) fitting the COMPoisson model can be relatively slow (and perhaps inaccurate) for highly overdispersed data (or, more precisely, for highly overdispersed conditional response). In particular, convergence of the iteratively reweighted least square algorithm depends on the accuracy of the approximations, so that ultimately this algorithm may not be best suited for fitting COMPoisson models with high overdispersion. However, it is easier to fit models on data indicative of under-dispersion ($\nu_{\text{CMP}} > 1$).

3.3 The main procedures in `spaMM`

3.3.1 Fitting functions

Five functions are available for fitting random-effect models (quick summary: use `fitme`, but refer to the documentation of `HLfit` and `HLCor` for some of its arguments):

`fitme` can fit models combining all the previously described features. This includes GLMMs, where the random effects are gaussian, but also models with non-gaussian random effects such as the Beta-binomial or negative-binomial (see examples below). For convenience, it can also fit LMs and GLMs.

⁴From the term $y \log(\lambda)$ of the log-likelihood, the link is $\eta = \log(\lambda(\mu))$, requiring the computation of $\lambda(\mu)$.

`fitmv` extends features of `fitme` to multivariate-response models (see section 4.4).

`corrHLfit` was previously implemented to fit spatial models. `fitme` is the recommended function now but `corrHLfit` should be fully functional. The two functions differ by the range of models fitted (`fitme` can fit both spatial and non-spatial models), by the names of some control arguments (`ranFix` and `etaFix` vs. the single argument `fixed`), by the default likelihood objective for fitting random-effect parameters (restricted likelihood for `corrHLfit`), and by the default algorithms used for maximization. `fitme`'s default methods can be much faster, in particular for large data sets when the residual variance model is a single constant term (no structured dispersion).

The `HLCor` function will provide fits of spatial models with given correlation parameters. Its default `method` is REML. Only for the CAR model it also allows estimation of the correlation parameter. For small datasets, it may then be faster than `fitme` (see further reference in Section 4.1.2), but it is otherwise of limited interest. However, the `HLCor` documentation is still relevant because `fitme` handles some arguments (the various correlation matrix specifiers, and `control.dist`) as if passed to `HLCor`.

`HLfit` is sufficient to fit non-spatial models, except families with an ad-hoc dispersion parameter (the two negative binomial families, `COMPOisson` and `beta_resp`) when this parameter must be estimated. Its default `method` is REML, so may still be used for a number of REML fits in documentation. One may not need to call it directly but the `HLfit` documentation is still relevant (notably for arguments `control.HLfit`, `rand.family`, and `verbose`) because all other fitting functions proceed as if calling it internally. It gives its name to the class of their returned objects (with “HL” for hierarchical likelihood). It is relatively slow for random-coefficient models; otherwise, for small datasets, it may be faster than `fitme`.

3.3.2 Post-fit functions

Post-fit inference functions notably include:

`confint`, to get provide confidence interval for a given parameter, either by parametric bootstrap, or (only for fixed-effects) by an asymptotic profile likelihood ratio.

LRT and `fixedLRT`, two slightly different interfaces to perform likelihood ratio tests of fixed effects. A bootstrap procedure is implemented to correct for small sample bias of the test. By use of offset terms, `fixedLRT` can also be contrived to provide more general profile likelihood ratio confidence regions.

Classical `anova` tables can also be computed. The `numInfo` function allows one to compute the information matrix for all parameters of a main-response model, and a similar procedure is used internally by `spaMM`'s `anova` method when applied to LMMs so that F -tests corrected by the “Satterthwaite method” (as developed by [Fai and Cornelius, 1996](#)) can be applied.

The `mapMM` and `filled.mapMM` functions provide colorful plots of the predicted response. The `predict`, `simulate`, and `update` methods extend the same-named procedures from the `stats` package; and extractor functions `logLik`, `fitted`, `fixef`, `ranef`, `vcov` (and so on) comparable to same-named functions from packages `stats` and `nlme/lmer`. `get_predVar` and related functions give variance of prediction and related quantities. The `get_inits_from_fit` function may be particularly useful to extract estimates from one fit in a form convenient to initiate another fit.

Diagnostic plots obtained by plotting the fitted object are shown in Fig. 2. Some are similar to those returned by a GLM fit, others would require more explanation. However, the most important point is that these plots are suspect, as they may suggest that the model is wrong when it is actually true (as can be verified by simulation for binary GLMMs or Poisson GLMMs with moderately large expected response values). The `DHARMA` package proposes more reasonable diagnostic plots, which can be produced from `spaMM` fit objects. However, the p-value which appears on one of them does not correspond to a generally valid goodness-of-fit test, yet appears to be widely overinterpreted as such a goodness-of-fit test in publications.

Of perhaps more notable interest are “partial-dependence” plots (Fig. 3), which evaluate the effect of a given fixed-effect variable, as (by default, the average of) predicted values on the response scale, over the empirical distribution of all other fixed-effect variables in the data, and of inferred random effects. This can be seen as the result of an experiment where specific treatments (given values of the focal variable) are applied over all conditions defined by the other fixed effects and by the inferred random effects. Thus, apparent dependencies induced by associations between predictor variables are avoided (see [Friedman, 2001](#), from which the name “partial dependence plot” is taken; or [Hastie et al., 2009](#), Section 10.13.2). This also avoids biases of possible alternative ways of plotting effects. In particular, such biases

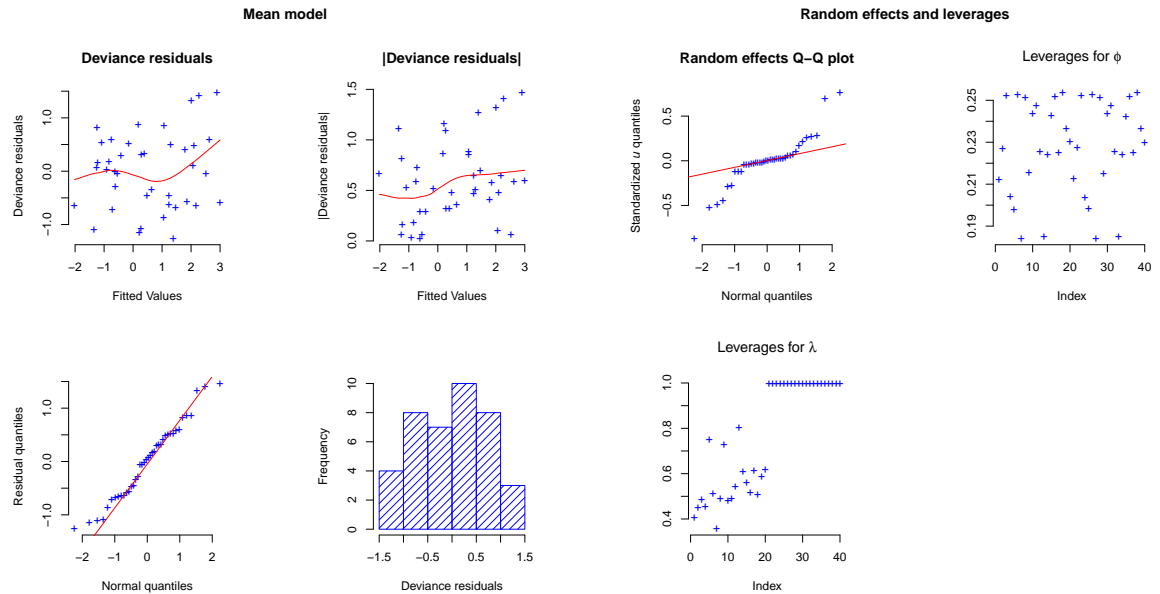


Figure 2: Diagnostic plots produced by `plot(HL1)`.

occur if the response link is not identity, and if averaging is performed on the linear-predictor scale or when other variables are set to some conventional value. As for other types of plots, `plot_effects` offers only a crude implementation of the concept using base graphic functions, but its source code can be adapted to your favorite graphic package (see also the `ggplot` example in the documentation of that function).

3.4 Multiple comparisons

A frequently requested feature is to perform multiple comparisons of means. It turns out that specifying `coef.=fixef.HLfit` makes `multcomp::glht` work on `spaMM` results, as in

```
library(multcomp)
set.seed(123)
irisr <- cbind(iris, id=sample(4, replace=TRUE, size=nrow(iris)))
irisfit <- fitme(Petal.Length ~ Species +(1|id), data=irisr,
                 family=Gamma(log))
summary(glht(irisfit, mcp("Species"="Tukey"), coef.=fixef.HLfit))
```

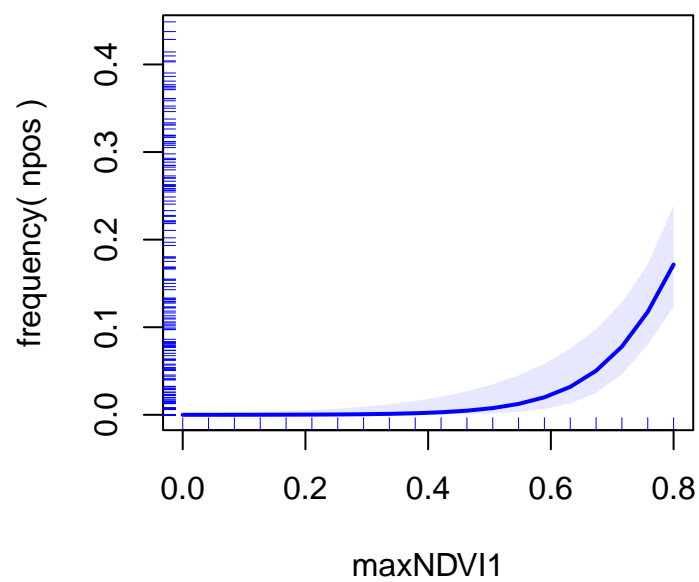


Figure 3: Partial-dependence plots produced by `plot_effects(lfit,"maxNDVI1")`.

3.5 Control of the fitting methods

This section first describes what is meant by the `method` argument of the fitting functions, and some further features of the REML method to guide (or not...) its use. It then tells a little about fitting algorithms, which can be controlled to some extent by initial-value arguments (but one rarely has to do that), and about controlling numerical methods for matrix computations, which can be useful if you are dealing with large data sets fitted by models involving unusual combinations of random effects.

3.5.1 Objective functions in ML, REML and PQL fits

The *starred* concepts mentioned in this Section are exemplified in Appendix A.

LMMs can be fitted by ML or REML, here in their standard meanings: ML maximizes (exact) marginal likelihood, and REML differs from ML as estimation of random-effect parameters and residual-dispersion parameters are obtained by maximizing (exact) restricted likelihood, which is the conditional likelihood of these parameters given (sufficient statistics for) the fixed effects. The inferred random effect values $\hat{\mathbf{v}}$, often known as best linear unbiased predictions (BLUPs), can be viewed as maximizing the *joint log-likelihood* of the data given the random effects, and of the random effects. The marginal likelihood is a “penalized” joint likelihood, that is, the joint likelihood minus a term deduced from the log-determinant of the matrix of second derivatives (*Hessian matrix*) of the negative joint likelihood with respect to the random effects (the negative Hessian is also the *observed information matrix*). The restricted likelihood (for REML estimation) is similarly evaluated, except that it uses the Hessian matrix with respect to random effects **and** fixed effects.

For GLMMs, exact marginal likelihood is generally too complex to evaluate and a *Laplace approximation* is widely used. The Laplace approximation to (log-)marginal likelihood takes the same form as marginal likelihood in LMMs, that is, the joint log-likelihood (or the * h -likelihood*, Lee et al., 2006) penalized by the log-determinant of the observed information matrix with respect to random effect values \mathbf{v} .

A further step taken in the h -likelihood literature is to approximate the estimated information matrix by its expectation under sampling of the response vector (for estimated parameter values). The two matrices are identical for GLM families with canonical link, but differ for non-canonical links (McCullagh and Nelder, 1989, p.42). This approximation has been used in past versions of `spaMM` and is still available as an option. But by default, `spaMM`

now use the observed information matrix. Thus, by default, the Laplace approximation used by **spaMM** is now identical to that used in packages based on automatic differentiation (Skaug and Fournier, 2006), such as **glmmTMB**, which also use the observed information matrix. The Laplace approximation with expected information can be used as the objective function by adding "exp" as a second specifier in `method=c("ML", "exp")` (or `c("REML", "exp")`).

For GLMMs, extensions of the computational techniques for REML corrections have long been considered (e.g., Breslow and Clayton, 1993). The definition of restricted likelihood from conditional likelihood does not generalize easily, but the concept of REML estimation can also be extended (a bit heuristically) as the maximization of joint log-likelihood penalized by the log-determinant of the information matrix with respect to random effects **and** fixed effects. REML, defined in this way (Lee et al., 2006, p. 187), has an effect analogous to its effect in LMMs, where it typically reduces the negative bias of ML estimates of dispersion parameters. Note that this extended definition still makes sense beyond GLM response families, i.e. beyond the cases considered in the literature.

The objectives functions considered here are variously described as penalized joint likelihoods, or as adjusted profile h -likelihoods (APHLs). The latter name has been retained in some of **spaMM**'s output as a generic name for all variants of approximate marginal or restricted likelihood. The (expected-Hessian) approximation for marginal likelihood is denoted $*p_v(h)*$ and the REML objective function is denoted $*p_{\beta,v}(h)*$. The observed-Hessian equivalents are denoted with a capital P: $P_v(h)$ and $P_{\beta,v}(h)$.

Various additional approximations have been considered. In penalized quasi-likelihood (PQL; e.g., Breslow and Clayton, 1993 and references therein), fixed effects are estimated by maximizing the joint likelihood rather than the Laplace approximation for marginal likelihood. As originally defined, it uses the extended restricted likelihood concept to estimate dispersion parameters. **spaMM** can perform both such fits ("PQL") and fits where dispersion parameters are estimated by maximization of the Laplace approximation for marginal likelihood rather than by REML ("PQL/L"). **spaMM** also implements some more cryptic methods discussed in the h -likelihood literature (see `help("method")` for details). The PQL results are clearly distinct from what `MASS::glmmPQL` produces (see Section 5.1). Although PQL has been criticized as an approximation of likelihood, it retains some interesting inferential properties, and can be much faster than the full Laplace approximation.

3.5.2 Implications of REML for post-fit inferences

By default, `fitme` (and `fitmv` for multivariate-response models) will fit all parameters jointly by ML. This default can be reversed by using `fitme(.,method="REML")`. Other fitting functions implemented in `spaMM` uses REML by default. This reflects some established practice for LMMs (e.g., in `lme4::lmer`), and also the fact that the other functions more closely follow by default the methods of Lee and Nelder, who provided the above generalized definition of REML, and generally emphasized REML adjustments (e.g., [Lee and Nelder, 1996](#), p. 633).

REML allow in principle more accurate estimation of the variance of random effects and of the residual variance. It may be useful for estimating other random-effect parameters. However, there are some drawbacks. For example, REML fits are not suitable for inference about fixed effects. Confidence intervals or likelihood ratio (LR) tests for fixed effects should be based on fits without REML estimation. The function `fixedLRT` may be useful to avoid errors here. It implements different procedures for inference about fixed effects, compared by [Rousset and Ferdy \(2014\)](#). For example, one can test for an effect of variable `pred` by using the `fixedLRT` function, whose arguments are similar to those of `fitme` but which takes one formula for each of the two models compared:

```
fixlrt <- fixedLRT(obs ~ 1 + Matern(1|x+y), obs ~ pred + Matern(1|x+y),
                  method="ML", data=d1, ranFix=list(nu=0.5))
summary(fixlrt,verbose=FALSE)

##      chi2_LR df      p_value
## 1 41.69444  1 1.067119e-10
```

For tests of random effect parameters, LR tests based on restricted likelihood are in principle more accurate, but likelihood ratios from full ML fits, possibly with some bootstrap correction, can also be performed. For computing LRTs for random effect parameters, no function equivalent to `fixedLRT` is available,⁵ so the tests must be performed by separately fitting the two models compared. Alternatively, confidence intervals can be computed using `confint`. CIs should always be considered as more informative summaries of the statistical analysis than LRTs of single null parameter values.

⁵A general-purpose function would have to account for the impact of constraints on parameter space on the distribution of the likelihood ratio, and for several other issues related to the various ways of specifying random effects, not fully characterized by a design matrix. By contrast, the space of linear regression coefficients is assumed unbounded, and nestedness of fixed-effect models is easily checked by operations on their design matrices.

It is also known that prediction based on REML fits can be less accurate than prediction based on ML fits (Harville, 1977). The estimation of spatial correlation parameters is not necessarily better by REML (Zimmerman, 2010).

3.5.3 Inner iterations versus outer optimization

`spaMM` combines the following methods to fit parameters of a model: (1) a (fairly standard) iteratively reweighted least squares (IRLS) algorithm to fit fixed-effect coefficients (β) and random-effect values (\mathbf{u}) given all correlation parameters (for LMMs, this reduces to the direct solution of a linear system); (2) an algorithm derived from the Levenberg-Marquardt one, to force convergence when the IRLS algorithm diverges; (3) generic function-optimization methods to fit correlation parameters such as the Matérn correlation parameters (generally using `nloptr`, but sometimes `minqa`); (4) for estimating dispersion parameters (λ s and ϕ s), either the same optimization methods, or iterative methods described in Lee et al. (2006). In the latter case, the joint estimation of β and of dispersion parameters is iterative: given some values at iteration t of these parameters, the IRLS algorithm is used to obtain new estimate of β and \mathbf{u} given the dispersion parameters, then new estimates of the dispersion parameters are obtained using the residuals of the current IRLS fit. In `spaMM`, the latter method is here called “inner iterative” to contrast it with the generic “outer optimization” method. The summary of a fit typically says which dispersion parameters were “outer-estimated”.

`HLfit` and `HLCor` only use the “inner” method while the other fitting functions may select either approach for estimating dispersion parameters. Both have virtues. The “inner” approach is always used to fit parameters of structured residual dispersion models. The inner approach may converge quickly and avoid being trapped in local maxima. It may have some occasional drawbacks when ϕ or λ should be fitted to zero. More importantly, it requires the computation of *leverages*, which is generally expensive, particularly for large datasets. So, a function such as `fitme` tends to select the “inner” method for small datasets and the “outer” one for large datasets (although the precise decision rule is based on additional criteria, as leverages are also often needed for evaluating the gradient of the Laplace approximation). By using the `init` or the `init.HLfit` arguments, one can control which method is used by `fitme` for specific parameters (see `help("inits")`).

These distinctions are also apparent in fits of a conditional autoregressive models, where the “inner” method may be used through `HLCor` (see examples in Section 4.1.2). This method uses an eigen-decomposition of the adjacency matrix, while the “outer” method can use a Cholesky factorization of the

precision matrix. For large adjacency matrices (say, roughly, of > 120 levels, although the size of the data matters too), the latter approach can take advantage of sparse Cholesky factorization to be faster. Consequently, the “inner” method is faster for small problems but `fitme` can be much faster otherwise.

There are also subtle differences in the REML fits achieved by `fitme`, `corrHLfit` and `HLCor`, due to differences in the optimization approaches. For example, `corrHLfit` will by default maximize with respect to correlation parameters the restricted likelihood of joint ML/REML fits of fixed-effect and dispersion parameters, while `HLCor` estimates all parameters by alternating ML estimation of fixed effects for given random-effect parameter estimates, and REML estimation of all random-effect parameters for given fixed-effects estimates. The differences can generally be ignored, except for detailed comparisons of algorithms and softwares (as could be illustrated using the example of Section 4.1.2).

3.5.4 Numerical methods

`spaMM` uses different matrix computations for three cases defined mainly from the structure of the correlation matrix of random effect and of its inverse, the precision matrix. These three cases are essentially (1) geostatistical models, which have dense correlation and dense precision matrix, which is why they are slow to fit; (2) conditional autoregressive or “Markov random field” (MRF) models, which have sparse precision matrices, whose sparsity can be exploited in fast algorithms. This has for example motivated the development of MRF approximations of the Matérn correlation model (Lindgren et al., 2011), further discussed in Section 5.2; (3) other non-spatial correlation models, which typically have sparse correlation matrices. The `how` function shows which method was selected for a fit. The default selection of method should be appropriate in most cases, but `help("algebra")` describes how to control it.

The correlation-based methods are based on a factorization of a weighted “augmented” model matrix akin to the ones described by Bates and DebRoy (2004) or Lee et al. (2006, p. 154). A QR factorization is used in `spaMM` except in some cases related to non-GLM response families, where the Cholesky factorization of the cross-product of the weighted augmented design matrix is used. The sparse-precision methods are not based on a factorization of the weighted augmented model matrix.

For LMMs with a single ϕ parameter, an “ y -augmented” version of the augmented model matrix is useful (Bates and DebRoy, 2004; Bates et al., 2015). Adaptations of this concept to the three above types of correlation

structures are implemented for such LMMs in `spaMM`. The `how` function also reports their use.

4 Further examples

4.1 Spatial GLMMs

4.1.1 Basic syntax

Non-Gaussian response data can be fitted by combining the features previously illustrated for LMMs together with syntax used in other procedures such as `glm` or `glmer`. For example, a geostatistical model for binomial data can be fitted by

```
data(Loaloe) ## parasite prevalence data in North Cameroon
binfit <- fitme(
  cbind(npos,ntot-npos) ~ 1 + Matern(1|longitude+latitude),
  data=Loaloe, family=binomial())
```

using the two-column response format `cbind(npos,ntot-npos)` for binomial data.

4.1.2 A classic example with autoregressive random effects

The following classical toy example (Clayton and Kaldor, 1987; Breslow and Clayton, 1993; Lee and Lee, 2012) considers a Poisson-distributed GLMM with a random effect following a conditional autoregressive (CAR) correlation model. The data describe lip cancer incidence in different Scottish districts (but we do not really care about the details). The model for the logarithm of expectation of the response is

$$\ln(\mu_i) = \ln(a_i) + \beta_1 + \beta_2 x_i/10 + b_i, \quad (4)$$

where $\ln(a_i)$ is an offset that describes the effect of population size and of some other variables, not included in the statistical model, on the Poisson mean; x_i is the variable `prop.ag` below; and b_i is a Gaussian random effect.

For the b_i s, the CAR model considers a covariance matrix of the form $\lambda(\mathbf{I} - \rho\mathbf{N})^{-1}$ where \mathbf{N} is an adjacency matrix between the different districts (a matrix with elements 1 if the districts are adjacent and 0 otherwise), here provided as `NMatrix` included in `data(scotlip)`. The rows of the matrix correspond to the `gridcode` variable in the data. A full fit including estimation of λ and ρ is then given by


```
data(scotlip)
lipfit <- HLCor(
  cases ~ I(prop.ag/10) + adjacency(1|gridcode) + offset(log(expec)),
  data=scotlip, family=poisson(), adjMatrix=Nmatrix)
```

We here use the `HLCor` function as it implements the same fitting strategy as that presumably used by [Lee and Lee \(2012\)](#), and the results are indeed very close to theirs:

```
summary(lipfit)

## formula: cases ~ I(prop.ag/10) + adjacency(1 | gridcode) + offset(log(expec))
## Estimation of lambda by REML (p_bv approximation of restricted logL).
## Estimation of fixed effects by ML (p_v approximation of logL).
## family: poisson( link = log )
## ----- Fixed effects (beta) -----
##               Estimate Cond. SE t-value
## (Intercept)      0.2377   0.2078   1.144
## I(prop.ag/10)    0.3763   0.1218   3.090
## ----- Random effects -----
## Family: gaussian( link = identity )
##               --- Correlation parameters:
##      1.rho
## 0.1740116
##               --- Variance parameters ('lambda'):
## Estimate of rho ( gridcode CAR):  0.174
## Estimate of lambda factor ( gridcode CAR):  0.1548
##               --- Coefficients for inverse[ lambda_i =var(V'u) ]:
##      Group      Term Estimate Cond.SE
## gridcode (Intercept)      6.46   1.716
## gridcode      adjd    -1.124   0.301
## # of obs: 56; # of groups: gridcode, 56
## ----- Likelihood values -----
##               logLik
## logL      (p_v(h)): -161.5141
## Re.logL   (p_b,v(h)): -163.6783
```

But this can also be fitted by `fitme`:⁶

⁶or by `corrHLfit`, with four-decimal differences in returned values due to differences in the strategies used to jointly maximize marginal and restricted likelihood (see Section 3.5.3).

```
data(scotlip)
lipfit <- fitme(
  cases ~ I(prop.ag/10) + adjacency(1|gridcode) + offset(log(expec)),
  data=scotlip, family=poisson(), adjMatrix=Nmatrix, method="REML")
```

HLCor will be faster for small data but `fitme` will otherwise be more efficient. See the `ohio.pdf` file on the [public repository for spaMM](#) for an example based on a comparison with the `hglm` package.

4.2 Beta random effects and binomial logit-Beta model

`spaMM` can now fit Beta-binomial models, using `family=betabin`. Alternatively, it can fit models with Beta-distributed random effects, which may also lead to a Beta-binomial model, but leads to a class of models that appears to often depart from the Beta-binomial models, as shown below. This class is here called the binomial logit-Beta model (per semantic considerations developed in a following long footnote), rather than beta-binomial HGLM as it is sometimes named. For concreteness, we assume that the response follows a binomial distribution with expectation p given conditionally on a realized random effect v

$$\text{logit}(p) = \ln \frac{p}{1-p} = \mathbf{x}\boldsymbol{\beta} + \mathbf{z}\mathbf{v} \quad (5)$$

(assuming the default logit link of the binomial GLM family); and we also assume that

$$v = \text{logit}(u) = \ln \frac{u}{1-u} \quad (6)$$

where the elements of u are independent Beta-distributed, and where the logit is also the default link for Beta-distributed random effects. Thus, if there are no fixed effects, the linear predictor gives $p = u$, again Beta-distributed. However, when a fixed effect a is included, $p = e^a u / [1 + (e^a - 1)u]$ no longer has a Beta distribution. This contrasts with Beta-binomial models where p is always Beta distributed, and where it is the (logit of the) expectation of the Beta variable which is controlled by a linear predictor (e.g., [Agresti, 2013](#)).

Since v is not gaussian, this is not a GLMM, but what [Lee and Nelder \(1996\)](#) called a hierarchical GLM (HGLM). It can be applied to a toy data set for Beta-binomial fits, of seed germination data. For comparison with the results of [Lee and Nelder \(1996\)](#), we can fit the model to these data by the method `HL(0,0)` (slightly cryptic at this step of the documentation), but otherwise the default method should be used.

```

data(seeds)
HLfit(cbind(r,n-r)~seed*extract+(1|plate),family=binomial(),
      rand.family=Beta(),method="HL(0,0)",data=seeds)

## formula: cbind(r, n - r) ~ seed * extract + (1 | plate)
## Estimation of lambda by REML (p_bv approximation of restricted logL).
## Estimation of fixed effects by h-likelihood approximation.
## family: binomial( link = logit )
## ----- Fixed effects (beta) -----
##
##              Estimate Cond. SE t-value
## (Intercept)      -0.54259   0.1864 -2.9102
## seed073           0.08003   0.3027  0.2644
## extractCucumber    1.33682   0.2643  5.0579
## seed073:extractCucumber -0.82202   0.4218 -1.9487
## ----- Random effects -----
## Family: Beta( link = logit )
##      --- Variance parameters ('lambda'):
## lambda = 4 var(u)/(1 - 4 var(u)) for u ~ Beta[1/(2*lambda),1/(2*lambda)];
##   plate : 0.02239
##      --- Coefficients for log(lambda):
## Group      Term Estimate Cond.SE
## plate (Intercept)  -3.799  0.5381
## # of obs: 21; # of groups: plate, 21
## ----- Likelihood values -----
##
##              logLik
##      h-likelihood: -42.16234
## logL      (p_v(h)): -54.00644
## Re.logL   (p_b,v(h)): -56.60452

```

The fixed effects estimates are consistent with those of [Lee and Nelder \(1996\)](#). The present parametrization of the Beta distribution is that of [Lee and Nelder \(2001\)](#) as discussed by [Lee et al. \(2006, p. 181\)](#), so that `HLfit`'s λ is $1/(2\alpha)$ for α as shown in [Lee and Nelder \(1996\)](#). The λ and α estimates are then seen to be approximately equivalent. [Lee and Nelder \(1996\)](#) also present a GLMM fit of these data, which is also similarly consistent with the GLMM fit by `HLfit` (not shown).

4.2.1 Gamma GLMM, HGLM, and joint GLMs

This example, derived from [Lee et al. \(2011\)](#), illustrates a Gamma GLMM model with a log link, that is $\boldsymbol{\eta} = \ln(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{v}$ where \mathbf{v} is normally

distributed.⁷ A notable feature is that it includes a non-trivial model for the variance of residual error, described by a linear predictor for the logarithm of this variance. There are only batch random effects (whose specification determine the elements of \mathbf{Z}), without any autocorrelated process, so the `HLfit` function is sufficient to analyze these data.

This example deals with data about semiconductor materials (“wafers”) from Robinson et al. Subject-matter details are ignored here; three variables denoted X_1 , X_2 and X_3 were experimentally varied. A fixed-effect model for the residual variance (“structured dispersion model”) was also considered. This model can be fitted by

```
data(wafers)
HLg <- HLfit( y ~ X1+X2+X3+X1*X3+X2*X3+I(X2^2)+(1|batch),
              family=Gamma(log),
              resid.model = ~ X3+I(X3^2) ,data=wafers)
summary(HLg)

## formula: y ~ X1 + X2 + X3 + X1 * X3 + X2 * X3 + I(X2^2) + (1 | batch)
## Estimation of lambda and phi by REML (P_bv approximation of restricted logL).
## Estimation of fixed effects by ML (P_v approximation of logL).
## family: Gamma( link = log )
## ----- Fixed effects (beta) -----
##              Estimate Cond. SE t-value
## (Intercept)  5.55720  0.05454 101.893
## X1           0.08376  0.02457   3.410
## X2          -0.20862  0.02400  -8.692
## X3          -0.13729  0.03745  -3.666
```

⁷ They call the Gamma GLMM with log link the Gamma-lognormal model. They appear to view this model as Gaussian $v = \ln(u)$ for u being lognormal, and to use the distribution of u as a basis for the name of the model (thus the “log” here comes from the $u \mapsto v$ link, not from the response link $\mu \mapsto \eta$). This terminology is ambiguous if the $u \mapsto v$ link and the u distribution are not both specified, as highlighted from the fact that the same distributions can be obtained with identity $u \mapsto v$ link if u has normal rather than lognormal distribution, in which case the name would be “Gamma-Normal” model.

This suggests that the semantics for HGLMs should be revised, and for example be based on the distribution of v so that different names for the same distribution cannot result from different specifications of u . In principle the link for the response should be specified although it is usually ignored when it is the canonical link of the GLM (which is not the case for the Gamma examples). According to this logic the Gamma-inverse Gamma model becomes the Gamma-log inverse Gamma (GLInG?) HGLM with log link, the not-quite Beta-binomial model (with canonical link for response) becomes the Binomial logit-Beta (BLoB?), and the usual Binomial GLMM becomes the Binomial logit-normal model (as in [Coull and Agresti, 2000](#)). A BLInG HGLM may not look like serious stuff, but it can be fitted...

```

## I(X2^2)      -0.07641  0.02038  -3.749
## X1:X3        -0.09181  0.03971  -2.312
## X2:X3        -0.08686  0.03969  -2.188
## ----- Random effects -----
## Family: gaussian( link = identity )
##      --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##   batch   : 0.02501
##      --- Coefficients for log(lambda):
##   Group      Term Estimate Cond.SE
##   batch (Intercept)  -3.688  0.4891
## # of obs: 198; # of groups: batch, 11
## --- Residual variation ( var = phi * mu^2 ) --
## Coefficients for log(phi) ~ X3 + I(X3^2) :
##      Estimate Cond. SE
## (Intercept) -2.8954  0.1384
## X3           0.1103  0.1142
## I(X3^2)      0.9464  0.1134
## ----- Likelihood values -----
##      logLik
## logL      (P_v(h)): -1157.607
## Re.logL (P_b,v(h)): -1175.199

```

A gamma-inverse Gamma model was also considered by [Lee et al. \(2011\)](#). Here the log of the expectation of the Gamma response has the form $\eta = \ln(\mu) = X\beta + \mathbf{v} = X\beta + \ln(\mathbf{u})$ where u has an inverse-Gamma distribution. v being non-Gaussian, this is an HGLM.

```

HLfit( y ~ X1+X2+X1*X3+X2*X3+I(X2^2)+(1|batch),
       family=Gamma(log),rand.family=inverse.Gamma(log),
       resid.model= ~ X3+I(X3^2) ,data=wafers)

## formula: y ~ X1 + X2 + X1 * X3 + X2 * X3 + I(X2^2) + (1 | batch)
## Estimation of lambda and phi by REML (P_bv approximation of restricted logL).
## Estimation of fixed effects by ML (P_v approximation of logL).
## family: Gamma( link = log )
## ----- Fixed effects (beta) -----
##      Estimate Cond. SE t-value
## (Intercept)  5.57053  0.05416 102.849
## X1           0.08373  0.02455   3.411
## X2          -0.20860  0.02399  -8.697
## X3          -0.13734  0.03744  -3.668
## I(X2^2)      -0.07637  0.02037  -3.750

```

```

## X1:X3      -0.09194  0.03970  -2.316
## X2:X3      -0.08682  0.03968  -2.188
## ----- Random effects -----
## Family: inverse.Gamma( link = log )
##      --- Variance parameters ('lambda'):
## lambda = var(u)/(1 + var(u)) for u ~ inverse-Gamma(sh=1+1/lambda, rate=1/lambda);
##   batch   : 0.02507
##      --- Coefficients for log(lambda):
## Group      Term Estimate Cond.SE
## batch (Intercept)  -3.686  0.4866
## # of obs: 198; # of groups: batch, 11
## --- Residual variation ( var = phi * mu^2 ) --
## Coefficients for log(phi) ~ X3 + I(X3^2) :
##      Estimate Cond. SE
## (Intercept)  -2.8967  0.1384
## X3           0.1093  0.1141
## I(X3^2)      0.9476  0.1134
## ----- Likelihood values -----
##      logLik
## logL      (P_v(h)): -1157.525
## Re.logL (P_b,v(h)): -1175.127

```

Lee et al. (2011) also fit GLMs, and GLMs with structured dispersion models (known as joint GLMs), to these data. These models can all be fit by HLfit. A joint GLM in particular is fit by

```

HLfit( y ~X1+X2+X1*X3+X2*X3+I(X2^2),family=Gamma(log),
       resid.model= ~ X3+I(X3^2) ,data=wafers)

## formula: y ~ X1 + X2 + X1 * X3 + X2 * X3 + I(X2^2)
## Estimation of phi by REML (p_bv approximation of restricted logL).
## Estimation of fixed effects by ML.
## family: Gamma( link = log )
## ----- Fixed effects (beta) -----
##      Estimate Cond. SE t-value
## (Intercept)  5.57570  0.03131 178.077
## X1           0.08375  0.02795   2.996
## X2          -0.21036  0.02795  -7.526
## X3          -0.13261  0.04019  -3.299
## I(X2^2)     -0.08017  0.02440  -3.286
## X1:X3       -0.09247  0.04383  -2.110
## X2:X3       -0.08201  0.04383  -1.871
## --- Residual variation ( var = phi * mu^2 ) --

```

```
## Coefficients for log(phi) ~ X3 + I(X3^2) :
##           Estimate Cond. SE
## (Intercept) -2.5119   0.1340
## X3          0.1588   0.1136
## I(X3^2)      0.7365   0.1125
## ----- Likelihood values -----
##                logLik
## logL           : -1170.187
## log restricted-lik : -1187.805
```

All these fits are by default REML fits: the argument `method="ML"` must again be used to perform ML fits. Results from these different fits of the same data are similar to published ones. These examples are continued in section 5.3.1 where we make sure that the small discrepancies with such estimates are not due to imprecisions of `spaMM`.

4.3 Random-slope model

A commonly considered random-slope model is a model with the following structure:

$$\boldsymbol{\eta} = \mathbf{1}\beta_I + \mathbf{b}_I + \mathbf{x}_S(\beta_S + \mathbf{b}_S). \quad (7)$$

The distinctive term is here $\mathbf{x}_S\mathbf{b}_S$ as the remainder is of the same form already considered e.g. in eq 4. The additional term means that the “slope” of the regression (the coefficient of the design variable \mathbf{x}_S) is random, including the random effect \mathbf{b}_S . Hence, there are two realized random effects $b_{I,g}$ and $b_{S,g}$ for each level g of the grouping factor. Random-slope models allow each such pair to be correlated, which is the main specificity in fitting these models. In this case the parameters of the random-slope term are thus described by a 2×2 correlation matrix.

The syntax for random-slope terms is `(<LHS>|<grouping factor>)`, where the left-hand-side `<LHS>` gives the explanatory variable \mathbf{x}_S , as in `HLfit(y ~X1+(X2|batch),data=wafers)`. If you want to ignore the correlation (which is often warned against), use two explicit terms as in `(1|batch)+(X2-1|batch)` or the shortcut `(X2||batch)`; if you further want a random effect on the slope only, consider only the term `(X2-1|batch)` or `(0+X2|batch)`; in all of these cases the syntax is the same as for a fit by `lmer` and is consistent with standard syntax for `formulas`.

Other possible forms of `<LHS>` (not all actually implying random-coefficient models) are detailed in `help("spaMM")`. They include the case where `<LHS>` is a factor variable, in which case for each level of the grouping factor, there are as many realized random effects as levels of the `<LHS>`. Thus, if the `<LHS>`

factor has three levels, a 3×3 correlation matrix describes parameters of the random-coefficient term.

The output from such models requires careful consideration. Suppose we fit

```
fitme(y ~X1+(X2|batch),data=wafers,method="REML")

## formula: y ~ X1 + (X2 | batch)
## REML: Estimation of ranCoefs and phi by REML.
##      Estimation of fixed effects by ML.
## Estimation of phi by 'outer' REML, maximizing restricted logL.
## family: gaussian( link = identity )
## ----- Fixed effects (beta) -----
##              Estimate Cond. SE t-value
## (Intercept)   224.91   12.316  18.262
## X1             23.68    9.325   2.539
## ----- Random effects -----
## Family: gaussian( link = identity )
##      --- Random-coefficients Cov matrices:
## Group      Term Var. Corr.
## batch (Intercept) 2760
## batch            X2 1844   -1
## # of obs: 198; # of groups: batch, 11
## ----- Residual variance -----
## phi estimate was 13064
## ----- Likelihood values -----
##              logLik
## logL          (p_v(h)): -1228.721
## Re.logL   (p_b,v(h)): -1222.139

## or HLfit(y ~X1+(X2|batch),data=wafers), but HLfit is much
## less efficient at fitting random-coefficient terms.
```

As in spatial models, correlated Gaussian random effects are represented as $\mathbf{b} = \mathbf{L}\mathbf{v}$ where the elements of \mathbf{v} are uncorrelated. The `Var.` column gives the variances of the *correlated* effects, $\mathbf{b} = \mathbf{L}\mathbf{v}$, which are also the random effects whose variance and standard deviation are reported by `lmer`'s `Variance` and `Std.Dev.`. The correlation coefficient for the “intercept” and “slope” effects is the `Corr` on the right of the random effect output (here as single -1 value; more generally a lower triangular block when more than two random effects are possibly correlated. By default, information about \mathbf{v} is not reported. It may be displayed by `summary(.,details=TRUE)`.⁸

⁸It is unclear how far such output would be useful because there is no unique repre-

4.4 Multivariate response

Multivariate-response models appear when different response variables depend on shared parameters, or on correlated random effects. A typical example is a quantitative-genetics model where different phenotypes are affected by an individual’s unobserved genotype. In some cases (requiring the same response family for all variables), these models can be fitted by previously described procedures, but when this is not feasible or convenient, the `fitmv` function should be used. It can fit models where the responses variables are affected by random effects identical or correlated across the variables. The case where variables are affected by the same fixed effect (i.e., by the same β coefficient for a given predictor variable) is not yet considered.

The `fitmv` function handles identical random effects over responses (say a shared `Matern` term), as well as random effects correlated across the response variables, specified by terms of the form `(mv(...) | <RHS>)`. The `mv(...)` specifier operates exactly as a factor, hence this term is a random-coefficient term as discussed in the previous section, but its levels refer to “levels” of the multivariate response (i.e., the different phenotypes, in the quantitative-genetics context), rather than to a factor variable in the `data`. This syntax has been extended in version 3.9.0 so that one can specify composite random effects that combines features of a `corrMatrix()` random effect and of the `mv()` specifier.

To further show the usage and relevance of these specifiers, we will reconsider an example previously taken by Wilson et al. (2010) to illustrate the use of different softwares in quantitative genetics. The next section will show how multivariate modelling can be used for hurdle models of species distribution (with additional discussion of prediction, simulation and bootstrap procedures for these models); and Appendix B provides further examples based on so-called “aster models” (Geyer et al., 2007). In their original form, the latter models have no shared random effects among response variables and can even be fitted by several calls to the `stats::glm` function. Nevertheless,

sentation of \mathbf{b} as \mathbf{Lv} . In the present case, the covariance matrix of \mathbf{b} can be represented in terms of its eigensystem, as $\mathbf{C_b} = \mathbf{L}\mathbf{\Lambda}\mathbf{L}'$ where \mathbf{L} contains normed eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. Thus $\mathbf{b} = \mathbf{Lv}$ where the variances of \mathbf{v} are these eigenvalues. Assigning these uncorrelated random effects to the intercept and the slope is a conceptually strained exercise: any given ordering of the eigenvectors in \mathbf{L} and for any permutation matrix \mathbf{P} , \mathbf{Lv} can be written as $(\mathbf{LP})(\mathbf{P}^\top \mathbf{v})$ in terms of the permuted design matrix \mathbf{LP} and permuted independent random effects $\mathbf{P}^\top \mathbf{v}$, so that each \mathbf{P} provides a statistically equivalent fit but a different assignment of \mathbf{v} elements to intercept and slope. However, `spaMM` chooses a permutation so as to maintain consistency between the output of models with and without correlation when the correlation vanishes, and to maintain consistency among the different descriptors of variance on each row in the same condition.

the response families may differ among variables, and `fitmv` is suitable to fit aster models or their mixed-model extensions.

Wilson et al. (2010) considered the joint modelling of birth weight (BWT) and tarsus length (TARSUS) in a simulated data set from a mythical Gryphon population. The quantitative genetic model assumes that genes which affect these traits have effects that are correlated among individuals according to a relatedness matrix, specified as a `corrMatrix` in `spaMM`. The multivariate model further assumes that genetic and environmental effects are correlated over traits within individuals.

A random effect (`0+mv(1,2)|ID`) can be used to represent correlated environmental effects. The `mv(...)` levels are the first response, birth weight, versus the second, tarsus length. The (`0+...`) syntax avoids contrasts being used in the incidence matrix of the random effects (consistently with the general syntax for factors in R), as it would not make much sense to represent TARSUS as a contrast to BWT. Another random effect `corrMatrix(0+mv(1,2)|ID)` can be used to represent genetic effects correlated over traits and individuals, with the appropriate correlation model for the multivariate quantitative genetic model (as detailed in `help("composite-ranef")`, the correlation matrix of such composite random effects can be written as the Kronecker product of the random-coefficient correlation matrix among levels of the factor by the correlation matrix specified by `corrMatrix`).

The `fitmv` function has a `submodels` argument which should be used to specify model formulas for each response. The function call

```
data("Gryphon")
gry_GE <- fitmv(
  submodels=list(
    BWT ~ 1 + corrMatrix(0+mv(1,2)|ID) + (0+mv(1,2)|ID),
    TARSUS ~ 1 + corrMatrix(0+mv(1,2)|ID) + (0+mv(1,2)|ID)
  ),
  fixed=list(phi=c(1e-6,1e-6)),
  corrMatrix = as_precision(Gryphon_A),
  data = Gryphon_df, method = "REML")

## Note: Precision matrix has 455 more levels than there are in the
data.
```

then fits the model. This results are practically identical to those obtained by the `ASreml` software (`help("Gryphon")` gives more details about this comparison).

`fitmv()` detects random effects terms shared accross different submodel formulas, so it detects that the same term `corrMatrix(0+mv(1,2)|ID)` af-

fects both responses, and likewise for $(0 + \text{mv}(1, 2) | \text{ID})$ (in this case this information is redundant with that given by the `mv()` specifier, but other types of shared random effects are detected in the same way: see Appendix B for examples). In the `Gryphon_df` data, measurements are not repeated within individuals, so the $(0 + \text{mv}(1, 2) | \text{ID})$ term also absorbs all residual variation. The residual variances must then be fixed to some negligible values (as shown) in order to avoid non-identifiability. The relatedness matrix `Gryphon_A` is here specified through an object that contains its inverse, using `as_precision()`. This is not required, but using it implies that `spaMM` does not have to find out and inform the user that using algorithms based on the inverse is better (as is typically the case for relatedness matrices).

4.5 Zero-altered (hurdle) models

Zero-altered, or hurdle, models combine a binomial model for non-zero versus zero response models, and a zero-truncated model for non-zero response. This is often contrasted to zero-inflated models, where the zeroes in the data are interpreted as a mixture of two events, such as a zero in a binomial response model plus a zero in a non-truncated Poisson response given the binomial response was non-zero. Zero-altered models can describe excesses or deficits of zeroes while zero-inflated models only describe excesses of zeroes. `spaMM` can fit the zero-altered models, but not the more constrained zero-inflated models (except by considering some quite inelegant use of offsets).

There is no specific syntax for hurdle models in `spaMM`. They can be fitted as two separate models for zero and non-zero response, or as a single multivariate-response model. Among other features, the multivariate-response fit can handle shared or correlated random effect among the two component models.

When conducting a parametric bootstrap inference, special care is needed in controlling the output of simulations from the two fitted submodels so that the simulations are appropriate for the question at hand.

To illustrate these features, we use abundance data for the common redstart (*Ph. phoenicurus*) in Switzerland. The data are part of a multispecies dataset discussed in several works on species distribution modeling, provided in the `AHMbook` package and reformatted in the `jSDM` package. We first define an ad-hoc data frame with both the counts `count` and the zero-truncated counts `Tobs`:

```
data("birds", package="jSDM")
redstart <- data.frame(
  x=birds$coordx/1000, y=birds$coordy/1000,
```

```

count=birds$`Phoenicurus phoenicurus`,
forest=birds$forest, elev=birds$elev, rlength=birds$rlength)

redstart$pres <- redstart$count>0

Tobs <- redstart$count
Tobs[Tobs==0L] <- NA
redstart$Tobs <- Tobs

```

4.5.1 Fitting

It is then possible to fit the (spatial) hurdle model as two separate models:

```

pres <- fitme(pres ~forest+elev+rlength + Matern(1|x+y),
  data=redstart, family=binomial())

## Fits using Laplace approximation may diverge for (nearly) all-
or-none binomial data:
## check PQL or PQL/L methods in that case.

Tabund <- fitme(Tobs ~forest+elev+rlength + Matern(1|x+y),
  data=redstart, family=Tpoisson())

```

The presence-absence model will typically be slower to fit than the zero-truncated Poisson one, due both to the typically much larger dataset and the fact that binary GLMMs are typically hard to fit.

The same models can also be jointly fit as a multivariate-response model using `fitmv`:

```

hurdle <- fitmv(
  submodels=list(
    list(pres ~forest+elev+rlength + Matern(1|x+y), family=binomial()),
    list(Tobs ~forest+elev+rlength + Matern(+1|x+y), family=Tpoisson())),
  data=redstart)

```

which is slower, as the fact that two independent models are being fit is being exploited by the fitting procedure. Note that the models are independent because (as explained in the previous section of multivariate-response models) the two `Matern` terms are recognized as distinct thanks to a little trick: one is declared as `Matern(1|x+y)` and the other as `Matern(+1|x+y)`, with the additional “+”.

Either way, it can be observed that the inferred spatial random effects are correlated among the two submodels:

```
Lv <- ranef(hurdle)
cov2cor(cov(cbind(Lv[[1]][names(Lv[[2]])], Lv[[2]])))

##           [,1]      [,2]
## [1,] 1.0000000 0.9191193
## [2,] 0.9191193 1.0000000
```

(where `Lv[[1]]`, the predictions for the random effect in the presence-absence model, is subsetting to positions `names(Lv[[2]])` where the other random effect has been predicted, using names determined by spatial locations). This observation may prompt us to fit as truly multivariate model with correlated random effects across the two submodels. Before performing this fit, we define a zero-truncation function whose immediate use in the model formula will be helpful in a later step:

```
ZT <- function(count, pres) {count[pres==0L] <- NA; count}

cor_hurdle <- fitmv(
  submodels=list(
    list(pres ~forest+elev+rlength + Matern(0+mv(1,2)|x+y), fam-
ily=binomial()),
    list(ZT(count,pres) ~forest+elev+rlength + Matern(0+mv(1,2)|x+y),
      family=Tpoisson()))),
  data=redstart)
```

Despite the additional cross-submodel correlation parameter this model has one fewer parameter than the previous one, as the same spatial correlation parameters are fitted for both responses. Despite this, it has a substantially better likelihood. The cross-correlation parameter is 1, at the boundary of parameter space, which suggests that an even better correlation model could be found, but otherwise confirms the preliminary observation, from the independent submodels, that the random effects were correlated. These fits also confirm the presence of spatial effects on redstart abundance (Guélat and Kéry, 2018).

4.5.2 Point-prediction intervals

Predictions from the fitted model by default use the fitted values stored in the fit object, which are dimensioned as the sum of number of observations

for each submodel. The same then holds for simulations. The length of `simulate(cor_hurdle)` is 266+61, because the dataset has 266 lines (for 266 locations), only 61 of which exhibit non-zero counts. By contrast, the length of `simulate(cor_hurdle, newdata=cor_hurdle$data)` (or of `simulate(., newdata=redstart)`) is 2*266, because predictions from the truncated Poisson model are then returned for locations where redstarts were not observed in the original data.

A third type of prediction/simulation may be needed in a bootstrap simulation aimed at taking into account all uncertainties in inference, as follows: (1) the binary model is simulated; (2) the truncated Poisson model is simulated for the locations simulated as occupied, that is generally neither the originally occupied locations nor all locations; (3) the model is refitted on these new data. Such simulations can be performed through a bit of programming.

Here, we illustrate another trick that may be useful to minimize such programming when a truly multivariate-response model is refitted: we have defined above the `ZT` function to allow the response of the zero-truncated model to be updated using new values of the `pres` and `count` variables, and we further need to specify that the `count` variable needs to be updated, because `spaMM` is unable to deduce this fact from the expression `ZT(count,pres)` for the response. We do this below by specifying `cor_hurdle$respNames <- c("pres","count")` (a message will direct you to the appropriate documentation if you forget to do so).

Further details of the bootstrap computation depend on what exactly we want to do. In particular, if we want to obtain intervals for expected numbers of observations (given the same observation effort) in the next year, and are willing to assume that the same values of latent factors (represented by the random effects) will be operating as in generating the data, then the new simulations should be conditional on the predicted values of the random effects (`type="residual"` in the following call).

```
newredstarts <-
  simulate4boot(cor_hurdle, newdata=redstart, type="residual",
               nsim=199L)$bootreps

cor_hurdle$respNames <- c("pres","count") # used by simfun() -> up-
date_resp()

simfun <- function(y, verbose=FALSE) {
  refit <- update_resp(cor_hurdle, newresp=y, verbose=verbose)
  repred <- predict(refit, newdata=refit$data)[,1]
```

```

    # again, using 'newdata' to get predictions for all locations.
    return(repred[1:266]*repred[267:532])
}

max_nb_cores <- parallel::detectCores(logical = FALSE) - 1L
library("doSNOW")
repreds <- spaMM_boot(
  boot_samples=newredstarts, nb_cores=max_nb_cores,
  simuland= simfun, fit_env= list(cor_hurdle=cor_hurdle, ZT=ZT),
  control.foreach=list(.errorhandling="stop"), type ="residual")$bootreps

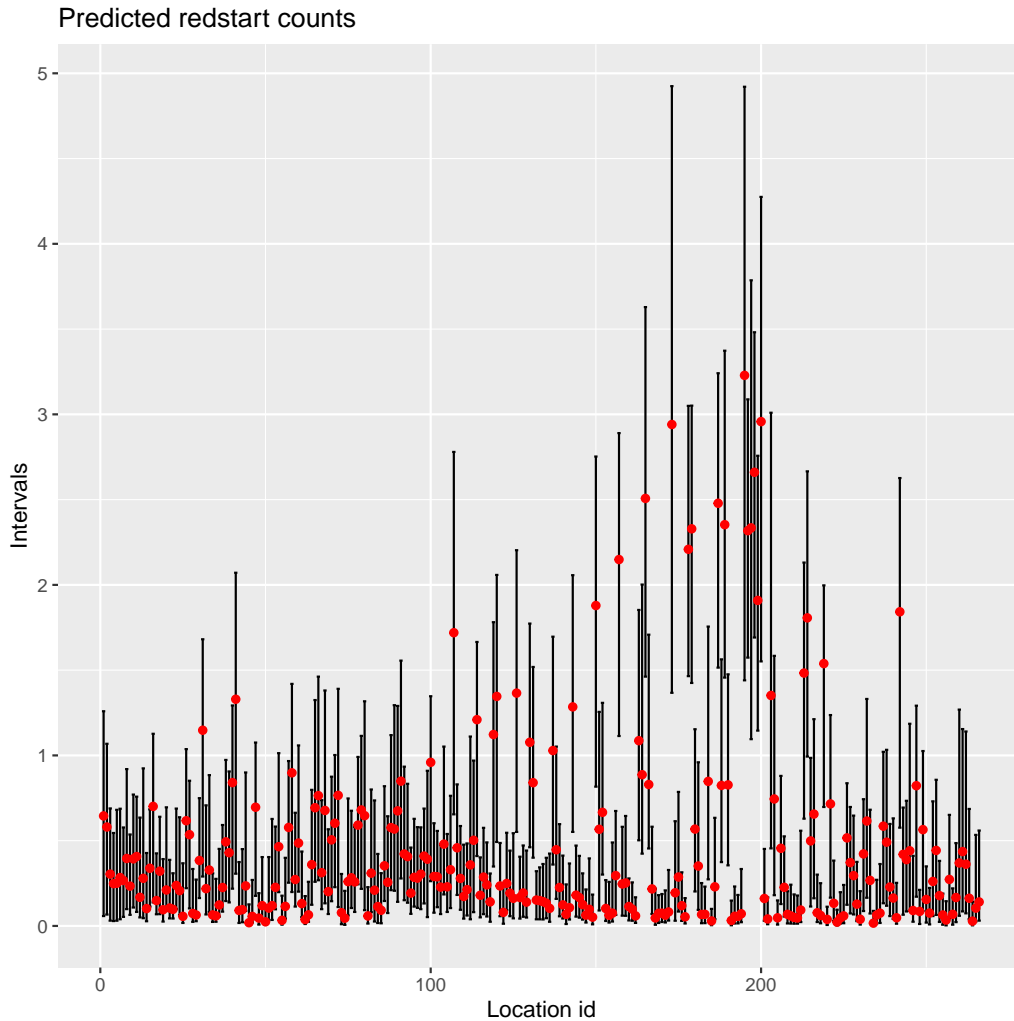
t0 <- predict(cor_hurdle, newdata=redstart)[,1]
t0 <- t0[1:266]*t0[267:532]

boot_interval <- function(idx) {
  boot::boot.ci(boot.out = list(t=repreds[,idx,drop=FALSE], t0=t0[idx],
    R=nrow(repreds), sim="parametric"), type=c("perc"))$percent[4:5]
}

CIs <- sapply(seq(ncol(repreds)), boot_interval)

```

After a rather long computation, the intervals for the successive locations in the data are (with point predictions as red dots):



5 Comparison with alternative software and literature results

This section brings two main types of informations: insights into the origin of often subtle differences between fit results by different procedures (up to the point of arguing about inconsistencies of summaries of `stats::glm` fits), and objective assessments of occasional claims about the relative speed of `spaMM` compared to `INLA`. Previous versions of this document presented a similar comparison with `hglm` for conditional autoregressive models, but this has been moved to another document (see further reference in Section 4.1.2). All

examples were computed with R Under development (unstable) (2023-08-21 r84998 ucrt), `spaMM` 4.4.0, and the given versions of the other packages.

5.1 Procedures for geostatistical models (or contrived to such usage): `MASS::glmmPQL`, `lmer`, `geoRglm`, `glmmTMB`

Some tricks commonly used (at least when `spaMM` was first conceived) to constrain the functions `lmer`, and `glmmPQL` (from `MASS`), to analyse geostatistical models are discussed in [Rousset and Ferdy \(2014\)](#) (in particular, in Appendix G, independently available [here](#)). In summary, they should be avoided. As further pointed below, `glmmPQL` does not really implement PQL, which is confusing.

Some packages based on stochastic algorithms (typically, MCMC) can fit spatial models, and can give reasonable results, but MCMC methods are typically difficult to assess, particularly in the absence of automated procedures for choosing Markov chain parameters. The same can be said about prior-laden approaches. Such software may also not provide procedures for LRTs of fixed effects.

One such package was `geoRglm` (now archived from CRAN). `spaMM` includes the `Loaloa` dataset, used by [Diggle and Ribeiro \(2007\)](#) and [Diggle et al. \(2007\)](#) in an application of methods implemented in `geoRglm`. The following call

```
fitme(cbind(npos,ntot-npos) ~ elev1 + elev2 + elev3 + elev4 +
      maxNDVI1 + seNDVI + Matern(1|longitude+latitude),
      data=Loaloa, family=binomial())
```

and additional examples in `help("Loaloa")` show how to obtain with `spaMM` results similar to previously published ones.

`glmmTMB` can fit the same spatial model, with equivalent results (`spaMM` is about 6.8 times faster than version 1.1.5, which means little except if you expected the opposite). Further comparisons of fitting times (and actual fitting success) were given by [Rousset and Courtiol \(2021\)](#) but are not up-to-date.

5.2 Interpolated Markov random fields via `spaMM` and INLA

For this comparison, and drawing on the `Loaloa` example above, we first define a `loa_spde` object that defines a particular Markov random field (MRF) structure for the random effects, which is designed to mimic a Matérn model with given smoothness, but to be faster to fit ([Lindgren et al., 2011](#)), because

it is designed to allow fast fitting by algorithms exploiting the sparseness of the precision matrix of the random effects. As **spaMM** also implements sparse-precision methods, it can also take advantage of this.

Based on this computational advantage many publications assume that fits of MRF models are automatically faster than those of Matérn models, but I could not find suitable comparisons, and the following ones show limits of this idea. The problem highlighted by true comparisons is that designing the MRF model involves the definition of a lattice which may involve more vertices than the number of locations in the original data, and that fitting an MRF model with many vertices is also slow. The number of vertices in the lattice can be reduced by using the **cutoff** argument of **INLA::inla.mesh.2d()**, but this may strongly affect the likelihood of the resulting fitted model, so this is not innocuous. Indeed, if the cutoff is high enough, some data points may be de facto excluded from the lattice, an event that seems to be largely ignored, but whose occurrence is reported by **spaMM**.

The **loa_spde** object describing the structure of the random effects is produced using functions from **INLA**:

```
spLoaloa <- sp::SpatialPointsDataFrame(
  coords = Loaloa[, c("longitude", "latitude")],
  data = Loaloa)
spde_mesh <- INLA::inla.mesh.2d(
  loc = INLA::inla.mesh.map(sp::coordinates(spLoaloa)),
  max.edge = c(3, 20))
loa_spde <- INLA::inla.spde2.matern(spde_mesh)
```

The random effect is then fitted by **spaMM**, using the syntax **IMRF(..., model=loa_spde)**. **INLA** is not required in this computation:

```
fitme(cbind(npos,ntot-npos)~
  elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
  + IMRF(1|longitude+latitude, model=loa_spde),
  family=binomial(), data=Loaloa)
```

The same random effect can be fitted by **INLA**, as follows:

```
fit_INLA <- inlabru::bru(
  components = npos ~ field(map = coordinates, model = loa_spde) +
  elev1+elev2+elev3+elev4+maxNDVI1+seNDVI,
  data = spLoaloa, family = "binomial", Ntrials=spLoaloa$ntot)
```

Computation times by INLA and `spaMM` are similar. This observation can be repeated on other examples (Rousset and Courtiol, 2021), even suggesting that `spaMM` is faster for large data sets.

The results of the lattice model by `spaMM` are indeed close to those of fitting the Matérn model with smoothness fixed to 1, i.e.

```
fitme(cbind(npos,ntot-npos)~
elev1+elev2+elev3+elev4+maxNDVI1+seNDVI+Matern(1|longitude+latitude),
fixed=list(corrPars=list("1"=list(nu=1))),
family=binomial(), data=Loaloe)
```

Comparison with the unconstrained Matérn model suggests that the latter is distinctly better at fitting these data, and fitting this unconstrained model is also faster than fitting the lattice model by either software. Similar observations can be repeated on other datasets involving similar numbers of spatial locations.⁹ In such cases, there is little motivation to use the lattice model defined by `INLA::inla.spde2.matern`, unless a `cutoff` is used to reduce the number of vertices in the lattice. Ultimately, such reduction (or a similar interpolation approach with a `Matern` random effect, not yet implemented) may be the only feasible one for large data sets. For datasets with thousands of positions, all methods will become very slow (again, unless a `cutoff` is used). A benefit of the lattice model that may matter in such cases is that it also requires less computer memory than the Matérn model.

5.3 Gamma GLMM with non-canonical link

We reconsider the previously introduced non-spatial Gamma GLMM, and some variations of it. The log link used here is standard, but non-canonical, and softwares may differ in the way this case is handled. The `HLfit` function (which performs REML fits by default) is used in this Section but identical results could be obtained using `fitme`.

HGLMM (Molas and Lesaffre, 2011) was previously considered in early versions of this documentation, but has been “removed from the CRAN repository” on 21/12/2013 (it is still available from the “archive”). In the example developed below, it gave exactly the same point estimates and likelihoods as the `HLfit` fit shown p. 28.

⁹Since this documentation was first written, the “unconstrained” IMRF model (declared as a `MaternIMRFa` random effect) has been introduced in `spaMM` and allows a better comparison with the unconstrained Matérn model.

5.3.1 A comparison with published estimates

The non-spatial Gamma GLMM fit considered here was considered by [Lee et al. \(2011\)](#), and the following analysis suggests that `spaMM` (and `HGLMMM`) are more accurate than the software used in that study (presumably `Genstat`). The likelihood values they give for this model are slightly higher than the `HLfit` ones but even higher than those that can be recomputed by `HLfit` for the estimates reported in the paper, which are given by¹⁰

```
phiGiven <- with(wafers,
  exp(as.matrix(cbind(1,X3,X3^2)) %*% matrix(c(-2.90,0.10,0.95))))
etaGiven <- with(wafers,
  5.55+0.08*X1-0.21*X2-0.14*X3-0.08*X2^2-0.09*X1*X3-0.09*X2*X3)
wafers <- cbind(wafers,etaGiven=etaGiven)
HLfit(y ~ (1|batch) + 0 + offset(etaGiven),
  family=Gamma(log),data=wafers,
  REMLformula=y ~X1*X3+X2*X3+I(X2^2)+(1|batch),
  ranFix=list(lambda=exp(-3.67),phi=phiGiven))

## formula: y ~ (1 | batch) + 0 + offset(etaGiven)
## family: Gamma( link = log )
## No fixed effect
## ----- Random effects -----
## Family: gaussian( link = identity )
## --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
## batch : 0.02548 [fixed]
## # of obs: 198; # of groups: batch, 11
## --- Residual variation ( var = phi * mu^2 ) ---
## phi was fixed to 0.128735 0.128735 0.128735 0.128735 0.157237 ...
## ----- Likelihood values -----
## logLik
## logL (P_v(h)): -1157.667
## (Non-standard?) ReL: -1175.267
```

Attempts to explain these discrepancies led to the following observations, beyond supporting the `HLfit` results. Discrepancies already occur in the fit of a simple Gamma GLM (still with log link), where `HLfit` computations

¹⁰this shows how to constrain `HLfit` fits using an offset. Another way is to use the `etaFix` argument (or fixed for function `fitme`), as `etaFix=list(beta=c("(Intercept)"=5.55, X1=0.08, X2=-0.21, X3=-0.14, "I(X2^2)"=-0.08, "X1:X3"=-0.09, "X2:X3"=-0.09))`. The `REMLformula` argument further allows to obtain the restricted likelihood, although no REML estimation is actually performed in this fit since no fixed-effect coefficient is estimated.

can easily be checked. In this case, the GLM weights being 1, the exact ML estimates of fixed effects are independent of the ϕ estimate, and it is easy to check that `HLfit` gives the same fixed effect estimates as `glm` does. Given known fixed effect estimates, the exact likelihood and exact restricted likelihoods are known functions of ϕ , and are also easily checked.

Such comparisons also highlight some subtleties with respect to dispersion estimation, which shows the more consistent behaviour of `spaMM` compared to `glm` in this respect. `method="ML"` will provide full ML estimates (exact for a GLM). This differs from the more confusing output of `glm`. From the displayed results of a `glm` fit, one can estimate ϕ as residual deviance/residual degrees of freedom, and this is the *approximate* REML estimate of ϕ given by `HLfit` with `method="EQL-"` or `method="RE(0,0,0)"`. However, this comparison is obscured by the idiosyncrasies of `summary.glm` (which returns an estimate of dispersion based on the Pearson residuals, not the deviance residuals). Further, `logLik` for `glm` objects does not return a likelihood comparable to those returned by the `method="EQL-"` fit, but rather the *approximate* marginal likelihood returned by `HLfit` with `method="ML(0,0,0)"`.

[Lee et al. \(2011\)](#) also considered a Gamma-inverse Gamma HGLM which is not implemented in all the above R packages. For this model `HLfit` and `GenStat` exhibit small discrepancies similar to those discussed above.

5.3.2 Further comparisons with `glmer`, and `glmmTMB`

Comparisons with `glmer` (from `lme4` version 1.1.33) were attempted, but it is not clear how to analyse a structured dispersion model (i.e., a model for the variance of the residual error) with `glmer`. Also it does not perform REML (in any extended definitions for GLMMs) for non-Gaussian response data. For comparison, we therefore first perform an ML fit without structured dispersion.

```
glmmfit <- fitme( y ~ X1*X3+X2*X3+I(X2^2)+(1|batch),
                 family=Gamma(log), data=wafers)

glmmfit

## formula: y ~ X1 * X3 + X2 * X3 + I(X2^2) + (1 | batch)
## Estimation of lambda and phi by ML (P_v approximation of logL).
## Estimation of fixed effects by ML (P_v approximation of logL).
## family: Gamma( link = log )
## ----- Fixed effects (beta) -----
##               Estimate Cond. SE t-value
## (Intercept)  5.61533  0.05622  99.874
## X1           0.08815  0.03366   2.619
```

```
## X3          -0.13903  0.03232  -4.302
## X2          -0.21165  0.03266  -6.480
## I(X2^2)     -0.10383  0.03258  -3.187
## X1:X3       -0.08992  0.04282  -2.100
## X3:X2       -0.08765  0.04280  -2.048
## ----- Random effects -----
## Family: gaussian( link = identity )
##          --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##   batch   : 0.01913
##          --- Coefficients for log(lambda):
##   Group      Term Estimate Cond.SE
##   batch (Intercept)  -3.956  0.5158
## # of obs: 198; # of groups: batch, 11
## --- Residual variation ( var = phi * mu^2 ) ---
## Coefficients for log(phi) ~ 1 :
##          Estimate Cond. SE
## (Intercept)  -1.833  0.1011
## Estimate of phi: 0.1599
## ----- Likelihood values -----
##          logLik
## logL      (P_v(h)): -1191.019
```

`glmmTMB` produces the same result. By contrast, the `lme4::glmer` fit provides slightly different parameter estimates but the maximized likelihood is substantially higher, which is intriguing. However, evaluation of the likelihood by numerical integration (which is straightforward given the simple structure of the random effects) shows that `spaMM`'s and `glmmTMB`'s approximations of the likelihood are more accurate than `glmer`'s one, and that they more closely maximize the true likelihood. In particular, numerical integration shows that the log likelihood is -1191.273 for the estimates given by `glmer`, which is distinct from `glmer`'s likelihood value, but very close to the value given by `fitme` with default method for the parameters estimates obtained with `glmer`:

```
merfit <- lme4::glmer( y ~ X1*X3+X2*X3+I(X2^2)+(1|batch),
                      family=Gamma(log), data=wafers)
logLik(merfit)

## 'log Lik.' -1188.672 (df=9)
```

```

wafers$X.beta <- predict(merfit, re.form=NA)
chk <- fitme( y ~(1|batch) + 0 + offset(X.beta),
             family=Gamma(log),data=wafers,
             fixed=list(lambda=VarCorr(merfit)$batch,
                        phi=sigma(merfit)^2))

logLik(chk)

##          P_v
## -1191.273

```

Numerical integration also shows that the likelihood is -1191.02 for parameters estimates given by `fitme`, which are therefore the better fit. The Laplace approximation based on the observed information matrix is $P_v = -1191.021$.

5.3.3 PQL vs. glmmPQL

`spaMM` implements further approximations, including [Breslow and Clayton's \(1993\)](#) PQL further discussed by [Lee and Nelder \(1996\)](#). See [Section 3.5.1](#) for an informal definition of PQL. `glmmPQL` is described as “equivalent to PQL up to details in the approximations” ([Venables and Ripley, 2002](#)), but the performance of `glmmPQL` is not a good guide to that of PQL ([Rousset and Ferdy, 2014, Appendix G](#)). One would have to dig into the `glmmPQL` code to understand the differences, which are already apparent in non-spatial models. E.g., one can compare the two following fits

```

data(wafers)
hfit <- fitme(y ~X1*X3+X2*X3+I(X2^2)+(1|batch),family=Gamma(log),
             data=wafers,method="PQL")
if(require(MASS,quietly = TRUE)) {
  gfit <- glmmPQL(y ~X1*X3+X2*X3+I(X2^2),random= ~ 1|batch,family=Gamma(log),
                 data=wafers)
}

```

The full output is not shown to save space, but e.g. ϕ estimates are 0.1648 vs 0.1508, and λ estimates are 0.02183 vs 0.01966 (`glmmPQL` may be closer to `spaMM`'s PQL/L method than to PQL). `glmmPQL` does not return likelihood values for comparison with `spaMM`'s ones.

5.4 Negative binomial model

The standard negative binomial model (with variance quadratically related to the mean) can be fitted using the `negbin2` family defined by `spaMM`:

```

fitme(cases~I(prop.ag/10)+(1|gridcode)
      +offset(log(expec)),data=scotlip,
      family=negbin2(),method="ML")

## formula: cases ~ I(prop.ag/10) + (1 | gridcode) + offset(log(expec))
## Estimation of fixed effects by ML (P_v approximation of logL).
## Estimation of lambda and NB_shape by 'outer' ML, maximizing logL.
## family: negbin2(shape=2.984)( link = log )
## ----- Fixed effects (beta) -----
##               Estimate Cond. SE t-value
## (Intercept)   -0.3528   0.1604  -2.199
## I(prop.ag/10)  0.7148   0.1479   4.834
## ----- Random effects -----
## Family: gaussian( link = identity )
##      --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##   gridcode : 2.032e-05
## # of obs: 56; # of groups: gridcode, 56
## ----- Likelihood values -----
##               logLik
## logL          (P_v(h)): -171.4703

```

Such fits are affected by the change in default approximation of likelihood effective with version 4 of **spaMM**. **negbin** can be used as a synonym for **negbin2** but **negbin2** is more clear now that the alternative **negbin1** model, with variance ‘linearly’ (affinely) related to the mean, is also implemented in **spaMM**.

A **negbin2** GLM can be represented as the Poisson-Gamma mixture model, namely as a “Poisson-log Gamma” model according to the semantics of footnote 7, with an individual-level Gamma-distributed random effect u and $v = \ln(u)$, such that the mean of the conditional Poisson response is of the form $\exp(\mathbf{X}\boldsymbol{\beta} + v) = u \exp(\mathbf{X}\boldsymbol{\beta})$, where $E(u) = 1$ and $\text{Var}(u) = 1/(\text{shape parameter of } \text{negbin})$. Thus, it can in principle be fitted using `family=poisson()` and `rand.family=Gamma(log)`. The shape parameter of **negbin** should then be compared to $1/\lambda$, the reciprocal of the variance of the Gamma random effects. This comparison may provide some insight into the performance of HGLM methods for non-gaussian random effects. However, the fit through such a mixed-model representation would use a Laplace approximation for the likelihood, in contrast to the GLM implementation which can exactly maximize likelihood. The Laplace approximation is less accurate for high variance of the individual-level Gamma random effect.

Acknowledgements

spaMM development has benefitted from a PEPS grant from CNRS and universities Montpellier 1 & 2, and from collaborations with Jean-Baptiste Ferdy and Alexandre Courtiol.

Bibliography

- Agresti, A. 2013. “Categorical data analysis,” Wiley, Hoboken, New Jersey, third edn.
- Bates, D., Mächler, M., Bolker, B., and Walker, S. 2015. Fitting Linear Mixed-Effects Models Using lme4, *Journal of Statistical Software* **67**, 1–48.
- Bates, D. M., and DebRoy, S. 2004. Linear mixed models and penalized least squares, *Journal of Multivariate Analysis* **91**, 1–17.
- Booth, J. G., and Hobert, J. P. 1998. Standard errors of prediction in generalized linear mixed models, *J. Am. Stat. Assoc.* **93**, 262–272.
- Breslow, N. E., and Clayton, D. G. 1993. Approximate inference in generalized linear mixed models, *J. Am. Stat. Assoc.* **88**, 9–25.
- Clayton, D., and Kaldor, J. 1987. Empirical Bayes estimates of age-standardized relative risks for use in disease mapping, *Biometrics* **43**, 671–681.
- Coull, B. A., and Agresti, A. 2000. Random effects modeling of multiple binomial responses using the multivariate binomial logit-normal distribution, *Biometrics* **56**, 73–80.
- Davison, A. C. 2003. “Statistical models,” Cambridge Univ. Press.
- Diggle, P., and Ribeiro, P. 2007. “Model-based geostatistics,” Springer series in statistics, Springer, New York.
- Diggle, P. J., Thomson, M. C., Christensen, O. F., Rowlingson, B., Obsomer, V., Gardon, J., Wanji, S., Takougang, I., Enyong, P., Kamgno, J., Remme, J. H., Boussinesq, M., and Molyneux, D. H. 2007. Spatial modelling and the prediction of *Loa loa* risk: decision making under uncertainty, *Ann. Trop. Med. Parasitol.* **101**, 499–509.

- Fai, A. H.-T., and Cornelius, P. L. 1996. Approximate F-tests of multiple degree of freedom hypotheses in generalized least squares analyses of unbalanced split-plot experiments, *Journal of Statistical Computation and Simulation* **54**, 363–378.
- Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine., *Ann. Statist.* **29**, 1189–1232.
- Geyer, C. J., Wagenius, S., and Shaw, R. G. 2007. Aster models for life history analysis, *Biometrika* **94**, 415–426.
- Guélat, J., and Kéry, M. 2018. Effects of spatial autocorrelation and imperfect detection on species distribution models, *Methods in Ecology and Evolution* **9**, 1614–1625.
- Harville, D. A. 1977. Maximum likelihood approaches to variance component estimation and to related problems, *J. Am. Stat. Assoc.* **72**, 320–338.
- Hastie, T., Tibshirani, R., and Friedman, J. 2009. “The elements of statistical learning: data mining, inference and prediction,” Springer, 2 edn.
- Lee, W., and Lee, Y. 2012. Modifications of REML algorithm for HGLMs, *Stat. Computing* **22**, 959–966.
- Lee, Y., and Nelder, J. A. 1996. Hierarchical generalized linear models, *J. R. Stat. Soc. B* **58**, 619–678.
- Lee, Y., and Nelder, J. A. 2001. Hierarchical generalised linear models: A synthesis of generalised linear models, random-effect models and structured dispersions, *Biometrika* **88**, 987–1006.
- Lee, Y., Nelder, J. A., and Park, H. 2011. HGLMs for quality improvement, *Applied Stochastic Models in Business and Industry* **27**, 315–328.
- Lee, Y., Nelder, J. A., and Pawitan, Y. 2006. “Generalized linear models with random effects: unified analysis via H-likelihood,” Chapman & Hall.
- Lindgren, F., and Rue, H. 2015. Bayesian Spatial Modelling with R-INLA, *Journal of Statistical Software* **63**, 1–25.
- Lindgren, F., Rue, H., and Lindström, J. 2011. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73**, 423–498.

- Matérn, B. 1960. “Spatial Variation: Stochastic models and their application to some problems in forest surveys and other sampling investigations,” Ph.D. thesis, Forest Research Institute, Stockholm, Sweden.
- McCullagh, P., and Nelder, J. A. 1989. “Generalized linear models,” Chapman & Hall, second edn.
- Molas, M., and Lesaffre, E. 2010. Hurdle models for multilevel zero-inflated data via h-likelihood, *Stat. Med.* **29**, 3294–3310.
- Molas, M., and Lesaffre, E. 2011. Hierarchical generalized linear models: The R package HGLMMM, *J. stat. Software* **39**, 1–20.
- Nychka, D., Bandyopadhyay, S., Hammerling, D., Lindgren, F., and Sain, S. 2015. A Multiresolution Gaussian Process Model for the Analysis of Large Spatial Datasets, *Journal of Computational and Graphical Statistics* **24**, 579–599.
- Rousset, F., and Courtiol, A. , 2021, [spaMM: an R package to fit generalized, linear, and mixed models allowing for complex covariance structures](#), Presented at the User2021 Conference.
- Rousset, F., and Ferdy, J.-B. 2014. Testing environmental and genetic effects in the presence of spatial autocorrelation, *Ecography* **37**, 781–790.
- Shmueli, G., Minka, T. P., Kadane, J. B., Borle, S., and Boatwright, P. 2005. A useful distribution for fitting discrete data: revival of the Conway–Maxwell–Poisson distribution, *appl. Stat.* **54**, 127–142.
- Skaug, H. J., and Fournier, D. 2006. Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models, *Comput. Stat. Data Anal* **51**, 699–709.
- Stein, M. L. 1999. “Interpolation of spatial data: some theory for Kriging,” Springer-Verlag, New York.
- Venables, W. N., and Ripley, B. D. 2002. “Modern applied statistics with S,” Springer-Verlag, New York, fourth edn.
- Wilson, A. J., Réale, D., Clements, M. N., Morrissey, M. M., Postma, E., Walling, C. A., Kruuk, L. E. B., and Nussey, D. H. 2010. An ecologist’s guide to the animal model, *Journal of Animal Ecology* **79**, 13–26.
- Zimmerman, D. L. 2010. Likelihood-based methods, in A. E. Gelfand, P. J. Diggle, M. Fuentes, and P. Guttorp (eds.), *Handbook of spatial statistics*, pp. 45–56, CRC Press, Boca-Raton (FL).

Appendices

A Evaluation of the likelihood approximations

An example of Gamma GLMM will here be used to detail the likelihood approximations, and some other concepts used in `spaMM`.

By default `fitme` estimates fixed-effect coefficients by maximizing the Laplace approximation $P_v(h)$ (`P_v`) for the log-likelihood. Laplace approximations $P_{\beta,v}(h)$ (`P_bv`) of the restricted log-likelihood may also be considered in REML fits. These approximations can be written

$$P_v(h) = h - 0.5 \ln \left\| -\frac{1}{2\pi} \frac{\partial^2 h}{\partial \mathbf{v}' \partial \mathbf{v}} \right\|. \quad (8)$$

and

$$P_{\beta,v}(h) = h - 0.5 \ln \left\| -\frac{1}{2\pi} \frac{\partial^2 h}{\partial (\boldsymbol{\beta}, \mathbf{v})' \partial (\boldsymbol{\beta}, \mathbf{v})} \right\|. \quad (9)$$

where h is the “joint log-likelihood”, and where for a given matrix \mathbf{H} , $\ln \|\mathbf{H}\|$ is the logarithm of the absolute value of its determinant (the “logdet”). The definitions and computation of h and of the distinct \mathbf{H} matrices involved in ML and REML fits will now be explained.

For concreteness we reconsider an example from the literature, fitting a Gamma GLMM to the `wafers` data, without the residual dispersion model considered by [Lee et al. \(2011\)](#). The default fit by `spaMM` is

```
byobs <- fitme( y ~ X1*X3 + X2*X3 + I(X2^2) + (1|batch),
family=Gamma(log), data=wafers, method=c("REML"))
logLik(byobs) # Restricted logL (given it's an REML fit), P_bv

##          P_bv
## -1207.473

logLik(byobs, which="P_v") # Marginal likelihood, P_v

##          P_v
## -1191.091
```

while the fit by [Lee et al.](#)'s method uses distinct likelihood approximations $p_{\beta,v}$ and p_v , and can be enforced by adding the second specifier “`exp`” in the `method` argument:

```

byexp <- fitme( y ~ X1*X3 + X2*X3 + I(X2^2) + (1|batch),
family=Gamma(log), data=wafers, method=c("REML","exp"))
logLik(byexp) # Restricted logL (given it's an REML fit), p_bv

##      p_bv
## -1207.51

logLik(byexp, which="p_v") # Marginal likelihood, p_v

##      p_v
## -1191.096

```

A.1 Conditional and joint log-likelihood

The log-likelihood for each independent draw y_k of a Gamma GLM can be written

$$c(\mu_k, \nu; y_k) = \nu \ln(\nu y_k / \mu_k) - \nu(y_k / \mu_k) - \ln(\Gamma(\nu)) - \ln(y_k) \quad (10)$$

where μ_k is the expected value (here conditional on the realized random effect), and $1/\nu$ is the variance of the residual term. Thus the conditional likelihood of the data given the realized random effects is

```

mui <- fitted(byobs)
nu <- 1/residVar(byobs, which="fit")
klik <- with(wafers, sum(nu*log(nu*y/mui)-nu*(y)/mui-log(gamma(nu))-log(y)))
klik

## [1] -1180.892

```

The joint log-likelihood is defined as the sum of this term and of the log-likelihood of the random effects:

$$h(\boldsymbol{\mu}, \nu, \lambda; \mathbf{y}, \mathbf{v}) = \sum_k c(\mu_k, \nu; y_k) + \sum_i \ln(L(v_i)) \quad (11)$$

where the sum over k is over all levels of the response variable and the sum over i is over all levels of the random effect.

The joint log-likelihood is called h -likelihood in the literature stemming from [Lee and Nelder \(1996\)](#). This includes [Molas and Lesaffre \(2010\)](#), [Lee et al. \(2011\)](#) and [Lee and Lee \(2012\)](#), upon which `spaMM` was initially designed. For this reason, the semantics and notation from these works are widely used

in `spaMM`, although the methods used in the current version of the package can depart in various ways from theirs. A capital P is here used to distinguish the Laplace approximations $P_v(h)$ and $P_{\beta,v}(h)$ from the ones used in earlier works. $p_v(h)$ and $p_{\beta,v}(h)$ are known as APHLs (adjusted profile h -likelihoods) in the h -likelihood framework. This acronym is used in `spaMM` where it may also include $P_v(h)$ and $P_{\beta,v}(h)$.

In the present example, the random effects are Gaussian with identity link, $u = v$, and dispersion λ :

$$\ln(L(v_i)) = -\frac{1}{2} \left(\frac{v_i^2}{\lambda} + \ln(2\pi\lambda) \right), \quad (12)$$

which may be recomputed as

```
ranefs <- ranef(byobs)[[1]]
lambda_obs <- VarCorr(byobs)[1,"Variance"] # or get_ranPars(byobs,"lambda")
hlik <- klik+with(byobs,sum(-(ranefs^2)/(2*lambda_obs)
-(log(2*pi*lambda_obs))/2))
hlik

## [1] -1173.463
```

Here the sum is over the 11 values of $v = \text{ranef}(\text{HLgs})[[1]]$.

`klik` and `hlik` are stored in the output object of the fit, and can be properly extracted by:

```
logLik(byobs, which="hlik")

##      hlik
## -1173.463

logLik(byobs, which="klik")

##      klik
## -1180.892
```

A.2 The gradient and Hessian matrix of conditional likelihood

The gradient is the vector of first-order derivatives, here with respect to the fixed-effect coefficients and random effects. Its elements can be written as

$$\frac{\partial c_i}{\partial \beta_p} = \frac{\partial c}{\partial \eta_i} \frac{\partial \eta_i}{\partial \beta_p} = \frac{\partial c}{\partial \eta_i} x_{ip} \text{ and likewise } \frac{\partial c_i}{\partial v_k} = \frac{\partial c_i}{\partial \eta_i} z_{ik} \quad (13)$$

in terms of elements of the design matrices \mathbf{X} and \mathbf{Z} .

The Hessian matrix is the matrix of second-order derivatives with respect to the same variables. Similar algebra shows that, e.g.

$$\frac{\partial^2 c_i}{\partial \beta_p \partial v_k} = \frac{\partial^2 c_i}{\partial \eta_i^2} x_{ip} z_{ik}. \quad (14)$$

In the Laplace all such derivatives are computed in β and v values maximizing the joint likelihood, and for the resulting value of η_i . Together with the derivative of the log-determinant, this ultimately yields the Laplace approximation, in terms of functions $\partial c / \partial \eta$ and $\partial^2 c / \partial \eta^2$ (third derivatives $\partial^3 c / \partial \eta^3$ may also be needed to evaluate the derivative of the log-determinant). This method is generic for all mixed-effect models defined in terms of a linear predictor η .

However, more specific expressions and approximations have been considered in the literature when additionally assuming that the response family is a GLM family (as is the case for the Gamma GLMM example). This leads in particular to the distinction between the default Laplace approximation P_v and the alternative approximation p_v . This distinction is detailed in the next Section.

A.3 Expected Hessian approximation for generalized linear models

We refer to standard notation for a GLM (McCullagh and Nelder, 1989, eq. 2.4). The likelihood of an observation is written in the form

$$L(y; \theta, \phi) = \exp\{[y\theta - b(\theta)]/a(\phi) + c(y, \phi)\}. \quad (15)$$

Three quantities are distinguished: θ , “the canonical parameter”, which is what factors with y ; μ , the expectation of y , and the linear predictor $\eta = \sum_j x_j \beta_j$. The assumed relationship between $\eta = g(\mu)$ defines the link g used, while the relationship between θ and μ defines the “canonical link”.

In a Gamma GLM, $a(\phi) = \phi = 1/\nu$, $\theta = -1/\mu$ (canonical link), $b(\theta) = \ln(\mu) = -\ln(-\theta)$ (McCullagh and Nelder, 1989, p. 30). The variance of Y is $b''(\theta)a(\phi) = \phi\mu^2$.

The elements of the gradient ∇l of the log-likelihood l can be written in the form

$$\frac{\partial l}{\partial \beta_p} = \frac{\partial l}{\partial \theta} \frac{\partial \theta}{\partial \eta} \frac{\partial \eta}{\partial \beta_p}. \quad (16)$$

We consider the Hessian matrix of l with respect to fixed-effect parameters, i.e. the matrix whose pr th element is $\partial^2 l / \partial \beta_p \partial \beta_r$. From eq. 16, it involves either $\partial^2 l / \partial \theta \partial \beta_r$, $\partial^2 \theta / \partial \eta \partial \beta_r$, or $\partial^2 \eta / \partial \beta_p \partial \beta_r$. The last term is null since η is linear, the second one is exactly null if the link is canonical ($\eta = \theta$), and the first one is $-\partial \mu / \partial \beta_r$.

If the link is not canonical, the second term is not null, so it should be either computed or approximated. The Hessian is often approximated by its expectation, given by

$$\mathbb{E} \left[\frac{\partial^2 l}{\partial \beta_p \partial \beta_r} \right] = \mathbb{E} \left[\frac{\partial^2 l}{\partial \theta \partial \beta_r} \frac{\partial \theta}{\partial \beta_p} + \frac{\partial l}{\partial \theta} \frac{\partial^2 \theta}{\partial \beta_p \partial \beta_r} \right]. \quad (17)$$

On random sampling of y for given μ , $\mathbb{E}[\partial l / \partial \theta] = \mathbb{E}[y - \mu] = 0$ hence the second term is null. Therefore, the second term of the expected Hessian is zero, and this can be used as an approximation for the equivalent term of the realized Hessian in the case of a non-canonical link.

A.4 Using observed versus expected Hessian

For fitting GLMs, a classical method (essentially the one implemented in the `stats::glm` function in R), iteratively modifies β estimates by $\Delta \beta = \mathbf{H}^{-1} \nabla l$ until the vector of parameter values where the likelihood gradient ∇l vanishes is reached. The same gradient-vanishing β is the target whether \mathbf{H} is the expected or observed Hessian.

In GLMMs, a similar fitting method can be used, but the gradient of likelihood is now a function of the Hessian matrix (through the gradient of its log-determinant with respect to fixed-effect coefficients), and thus the likelihood approximation and the parameter estimates should differ whether the gradient computation is based on the expected (approximate) Hessian or the observed Hessian, when the two matrices differ. The observed Hessian leads to a Laplace approximation P_v , while only the expected Hessian matrix has been considered in the h -likelihood literature, leading to the likelihood approximation p_v . Similar distinctions holds for restricted likelihood.

A major drawback of the expected Hessian approximation is that it cannot be applied (at least practically) for non-GLM response families. With the introduction of such response families in `spaMM`, methods based on the observed Hessian were introduced. As various confusions would likely result from using different default approximation methods for different response families, the likelihood approximations using the observed Hessian are now the default ones for all response families, whether GLM ones or not.

This distinction also has implications in terms of quality of approximation of likelihood, of convergence of iterative algorithms, and of implementation of

matrix computations. For example, using the observed Hessian often allows faster fits for negative binomial GLMMs. Again, it makes no difference for models with canonical link, such as binomial(logit) or poisson(log). For the Gamma response family the log link is not canonical, which explains the difference between P_v and p_v .

A.5 Hands-on computation of Laplace approximations

We can first compute the elements of the matrices whose “logdet” is required.¹¹

Generalizing from the gradient expression for GLMs, we can write

$$\frac{\partial h}{\partial \beta_p} = \nu \sum_{ij} (y_{ij} - \mu_{ij}) w_{ij} \frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial \eta_{ij}}{\partial \beta_p} \quad (18)$$

$$\frac{\partial h}{\partial v_k} = \nu \sum_{ij} (y_{ij} - \mu_{ij}) w_{ij} \frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial \eta_{ij}}{\partial v_k} - \frac{v_k}{\lambda} \quad (19)$$

where $\eta_{ij} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 + \beta_{22} x_2^2 + z_{ij,k} v_k$ is the linear predictor for the “mean” part and w_{ij} is the diagonal element of the weight matrix

$$\mathbf{W}_\phi \equiv \text{diag}(w_{ij}) \text{ where } w_{ij} = \left(\frac{\partial \mu_{ij}}{\partial \eta_{ij}} \right)^2 / b''(\theta_{ij}). \quad (20)$$

Here with a log link, $\partial \mu / \partial \eta = \partial \mu / \partial \ln(\mu) = \mu$, $w_{ij} = 1$ and $\mathbf{W}_\phi = \mathbf{I}$.

The design matrix for random effects has elements here denoted $z_{ij,k}$ for observation ij and column k . $z_{ij,k}$ is the indicator that observation ij belongs to batch k , hence $z_{ij,k} = \delta_{\text{batch}(k), i}$. The design matrix for fixed effects has elements here denoted $x_{ij,p}$ for coefficient p of the fixed effects.

We consider a log link ($\eta = \ln(\mu)$), so the link is not canonical ($\eta \neq \theta$) and we first consider the expected Hessian approximation to the observed Hessian, as explained in Section A.3. In particular for the derivatives with respect to β_r of the different factors in (18), we again note that the last factor has a null derivative, and that the derivative of the middle ones can be ignored when the $(y - \mu)$ is approximated by its null expectation. Thus

¹¹Usefully detailed computations are also presented by [Molas and Lesaffre \(2010\)](#) for a Poisson “hurdle” HGLM. Our computations differ from theirs as we have only one random effect in the mean part and the model is Gamma (with a dispersion parameter $\phi \equiv 1/\nu$) rather than Poisson (without any overdispersion parameter). We do not need their correction term $M(\theta_{ijk})$ for truncation of the Poisson distribution.

we only consider the remaining derivative

$$\frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial (y_{ij} - \mu_{ij})}{\partial \beta_r} = -\frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial \mu_{ij}}{\partial \beta_r} = -\frac{\partial \eta_{ij}}{\partial \beta_r} = -x_{ij,r}. \quad (21)$$

Hence the nonzero elements of the expected Hessian matrix are

$$E \frac{\partial^2 h}{\partial \beta_p \partial \beta_r} = -\nu \sum_{ij} x_{ij,p} \frac{\partial \eta_{ij}}{\partial \beta_p} = -\nu \sum_{ij} x_{ij,p} x_{ij,r}, \quad (22)$$

$$E \frac{\partial^2 h}{\partial \beta_p \partial v_k} = -\nu \sum_{ij} x_{ij,p} z_{ij,k}, \text{ and} \quad (23)$$

$$E \frac{\partial^2 h}{\partial v_k^2} = -\nu \sum_{ij} z_{ij,k}^2 - 1/\lambda. \quad (24)$$

Only the second derivatives with respect to the v_k 's are involved in the approximation to the marginal likelihood of the fitted model, which is

$$p_v(h) = h - 0.5 \ln \left\| -\frac{1}{2\pi} \frac{\partial^2 h}{\partial \mathbf{v}' \partial \mathbf{v}} \right\|. \quad (25)$$

which we can reconstruct as

```
designZ <- get_ZALMatrix(byexp)
lambda_exp <- VarCorr(byexp)[1,"Variance"]
nu_exp <- 1/residVar(byexp, which="fit")
tZZ <- crossprod(designZ)
hess <- - nu_exp*tZZ -diag(1/lambda_exp,11) # expected Hessian matrix
logdet <- determinant(hess/(2*pi))$modulus[[1]]
# = sum(log(diag(-hess)/(2*pi))) here
(p_v <- logLik(byexp,which="hlik")[[1]]-logdet/2) # = logLik(byexp,"p_v")

## [1] -1191.096
```

By contrast the diagonal elements of the observed Hessian matrix are $\partial^2 h / \partial v_k^2 = -\nu \sum_{ij} z_{ij,k}^2 y_{ij} / \mu_{ij} - 1/\lambda$ and then P_v is reconstructed as

```
lambda_obs <- VarCorr(byobs)[1,"Variance"]
nu_obs <- 1/residVar(byobs, which="fit")
d2logclik_deta2 <- drop(byobs$y/fitted(byobs))
weights_obs <- nu *
  crossprod(designZ, diag(d2logclik_deta2) %*% designZ)
# observed Hessian matrix:
```

```

hess <- - weights_obs -diag(rep(1/lambda_obs,11))
logdet <- determinant(hess/(2*pi))$modulus[[1]]
(P_v <- logLik(byobs,which="hlik")[[1]]-logdet/2) # = logLik(byobs,"P_v")

## [1] -1191.091

```

Likewise the restricted log-likelihood approximations $p_{\beta,v}$ and $P_{\beta,v}$ can be reconstructed as follows :

```

designX <- get_matrix(byobs)
# The joint design matrix for fixed and random effects:
Xa <- cbind(designX,designZ)

# p_bv
tXXa <- crossprod(Xa)
hess <- - nu*tXXa -diag(c(rep(0,7),rep(1/lambda_exp,11)))
logdet <- determinant(hess/(2*pi))$modulus[[1]]
(p_bv <- logLik(byexp,which="hlik")[[1]]-logdet/2) # = logLik(byexp)

## [1] -1207.51

# P_bv
weights_obs <- nu_obs*
  crossprod(Xa, diag(drop(byobs$y/fitted(byobs))) %*% Xa)
hess <- - weights_obs -diag(c(rep(0,7),rep(1/lambda_obs,11)))
logdet <- determinant(hess/(2*pi))$modulus[[1]]
(P_bv <- logLik(byobs,which="hlik")[[1]]-logdet/2 ) # = logLik(byobs)

## [1] -1207.473

```

B Multivariate analyses: the aster example

The **aster** package has been developed to fit joint GLMs for multivariate responses with different response families. The basic example from the **aster** package considers a life-history dataset recording survival, flowering and number of flower heads of 570 *Echinacea angustifolia* individuals over three years (2002 to 2004). In its original form this example has no random effects and **glm** fits could as well be used (as we will see), but we will add shared random-effects for our purposes.

Data preparation for analysis by **aster** reads as

```

library(aster)
data(echinacea)
vars <- c("ld02", "ld03", "ld04", "fl02", "fl03", "fl04",
"hdct02", "hdct03", "hdct04")
redata <- reshape(echinacea, varying = list(vars), direction = "long",
timevar = "varb", times = as.factor(vars), v.names = "resp")
redata <- data.frame(redata, root = 1)
hdct <- grepl("hdct", as.character(redata$varb))
redata <- data.frame(redata, hdct = as.integer(hdct))
level <- gsub("[0-9]", "", as.character(redata$varb))
redata <- data.frame(redata, level = as.factor(level))

```

An individual cannot flower (fl variables) if it has not survived the previous year (ld variables), and there is no head count (hdct variables) to analyze for individual plants that did not produce flowers. `aster`'s way of handling that is not by specifying missing data as NA. Instead, these data are coded as 0 and `aster` handles logical dependencies between variables by the `pred` argument that gives the predecessor of a variable in a directed acyclic graph. So the `aster` fit itself is achieved by

```

pred <- c(0, 1, 2, 1, 2, 3, 4, 5, 6)
fam <- c(1, 1, 1, 1, 1, 1, 3, 3, 3) # response families
aout_cond <- aster(resp ~ varb + level : (nsloc + ewloc) + hdct : pop,
pred, fam, varb, id, root, data = redata, type="conditional")

```

Its results are consistent to those of three GLM fits with missing data coded as NA. We can define a function to implement this coding:

```

asNA.acyclic <- function(data, pred) {
order_pred <- order(pred)
predvars <- names(pred)
ord_desc_vars <- names(pred[order_pred])
for (ordered_it in order_pred) {
descvar <- ord_desc_vars[ordered_it]
predecessor <- pred[descvar]
if (predecessor > 0L) {
predvar <- predvars[predecessor]
# Predecessor variable being 0 (no survival, or no flowers) means
# that descendant variable cannot be observed:
data[is.na(data[[predvar]]) | data[[predvar]] == 0L, descvar] <- NA
}
}
return(data)
}

```

```

}
# Note the names, important here:
varpred <- c(ld02=0, ld03=1, ld04=2, fl02=1, fl03=2, fl04=3,
hdct02=4, hdct03=5, hdct04=6)
NAechin <- asNA.acyclic(echinacea, pred=varpred)

```

and fits by `glm` with missing data can then be performed after a little more reshaping. E.g.

```

recond <- reshape(NAechin, varying = list(vars), direction = "long",
timevar = "varb", times = as.factor(vars), v.names = "resp")
recond$varld <- recond$varfl <- recond$varhdct <- recond$varb
recond$varld[ ! grepl("ld", as.character(redata$varb))] <- NA
recond$varfl[ ! grepl("fl", as.character(redata$varb))] <- NA
recond$varhdct[ ! grepl("hdct", as.character(redata$varb))] <- NA
ld_fit_by_glm <- glm(resp ~ varld + nsloc + ewloc, family=binomial(), data=recond)
# *etc.*

```

and two other GLMs, but we skip the details as further comparisons will be more conveniently performed on the fit by the `fitmv` function:

```

(spast <- fitmv(submodels=list(
has_survived=list(resp ~ varld + nsloc + ewloc, family=binomial()),
has_flowers=list(resp ~ varfl + nsloc + ewloc, family=binomial()),
head_count=list(resp ~ varhdct + nsloc + ewloc + pop, family=Tpoisson()),
data=recond))

## formula_1: resp ~ varld + nsloc + ewloc
## formula_2: resp ~ varfl + nsloc + ewloc
## formula_3: resp ~ varhdct + nsloc + ewloc + pop
## Estimation of fixed effects by ML.
## Families: 1: binomial( logit ); 2: binomial( logit ); 3: 0-truncated poisson( log )
## ----- Fixed effects (beta) -----
##          Estimate Cond. SE t-value
## (Intercept)_1  1.035694 0.099716 10.3865
## varldld03_1    1.996676 0.250670  7.9654
## varldld04_1    2.299822 0.290504  7.9167
## nsloc_1        0.081380 0.012462  6.5301
## ewloc_1        0.020498 0.012543  1.6343
## (Intercept)_2 -0.557288 0.105435 -5.2856
## varflfl03_2    -0.312861 0.152042 -2.0577
## varflfl04_2    0.771270 0.149318  5.1653
## nsloc_2        0.062696 0.009277  6.7585
## ewloc_2        0.037360 0.009224  4.0504
## (Intercept)_3  0.526213 0.153222  3.4343
## varhdcthdct03_3 0.036662 0.121748  0.3011
## varhdcthdct04_3 0.557703 0.095751  5.8245
## nsloc_3        -0.008575 0.006783 -1.2641
## ewloc_3        0.012434 0.006545  1.8996
## popEriley_3    -0.568350 0.170541 -3.3326
## popLf_3        -0.586309 0.186966 -3.1359
## popNWLf_3      -0.073558 0.150309 -0.4894
## popNessman_3   -0.222755 0.259245 -0.8592
## popSPP_3       -0.185061 0.155933 -1.1868
## popStevens_3   -0.151462 0.163319 -0.9274
## ----- Likelihood values -----
##          logLik
## logL       : -1920.922

```

In the output, each fixed-effect coefficient is indexed by the submodel to which it belongs. These coefficients are the same as those returned by `glm` on

each submodel. For the comparison to the `aster` results, the `type="conditional"` argument we used in the `aster` call turns out to be important (see the `aster` documentation for details, but retain here that we will compare outputs consistent with those of GLMs fitted in a standard way in R). The different model fits may still be difficult to compare, as `aster` used a different sets of contrasts for the fixed effects, but we can more easily compare predictions of the conditional models, here for the fits without random effects:

```
spaMM_pred <- predict(spast) []

## NA's in required variables: prediction not possible for all 'new-
## data' rows.
## NA's in required variables: prediction not possible for all 'new-
## data' rows.
## NA's in required variables: prediction not possible for all 'new-
## data' rows.

aster_pred <- predict(aout_cond, model.type="conditional")
range(c(0,sort(unique(spaMM_pred)))-sort(unique(aster_pred)))

## [1] -6.707523e-12  3.600364e-11
```

the only difference between the predictions is the additional “0” predicted value by `aster` in cases where `spaMM` does not return predictions (for missing response values in the data).

Instead of reshaping the data to 9×570 rows, is is possible to reshape by year (3×570 rows):

```
yearvars <- c("ld02", "ld03", "ld04")
byyear <- reshape(NAechin, varying = list(yearvars), direction = "long",
timevar = "varld", times = as.factor(yearvars), v.names = "respld")
yearvars <- c("fl02", "fl03", "fl04")
flbyyear <- reshape(NAechin, varying = list(yearvars), direction = "long",
timevar = "varfl", times = as.factor(yearvars), v.names = "respfl")
yearvars <- c("hdct02", "hdct03", "hdct04")
hdctbyyear <- reshape(NAechin, varying = list(yearvars), direction = "long",
timevar = "varhdct", times = as.factor(yearvars), v.names = "resphdct")
# Combine results of each reshape in one data.frame:
byyear$varfl <- flbyyear$varfl
byyear$varhdct <- hdctbyyear$varhdct
byyear$respfl <- flbyyear$respfl
byyear$resphdct <- hdctbyyear$resphdct
byyear <- byyear[,c(12,4:6,10,13,14,11,15,16)]
head(byear)
```

##	id	pop	ewloc	nsloc	varld	varfl	varhdct	respld	respfl	resphdct	
##	1.ld02	1	NWLF	-8	-11	ld02	fl02	hdct02	0	NA	NA
##	2.ld02	2	Eriley	-8	-10	ld02	fl02	hdct02	1	0	NA
##	3.ld02	3	NWLF	-8	-9	ld02	fl02	hdct02	0	NA	NA
##	4.ld02	4	SPP	-8	-8	ld02	fl02	hdct02	0	NA	NA
##	5.ld02	5	SPP	-8	-7	ld02	fl02	hdct02	0	NA	NA
##	6.ld02	6	Eriley	-8	-6	ld02	fl02	hdct02	1	0	NA

as used in the next fits.

To assess whether an individual random effect affected both survival and flowering, we can fit

```
asMM <- fitmv(submodels=list(
  list(respld ~ varld + nsloc + ewloc+(1|id), family=binomial()),
  list(respfl ~ varfl + nsloc + ewloc+(1|id), family=binomial()),
  list(resphdct ~ varhdct + nsloc + ewloc+pop, family=Tpoisson())),
data=byyear)
```

The crucial syntactic feature here is that identical random-effect terms across submodels (here (1|id)) are recognized as a single random effect.

Given that `nsloc` and `ewloc` are spatial coordinates, we can also fit a spatial random effect. Overall, a mixed-effect version of the `aster` fit may be

```
(spasMM <- fitmv(submodels=list(
  list(respld ~ varld + Matern(1|nsloc + ewloc), family=binomial()),
  list(respfl ~ varfl + Matern(1|nsloc + ewloc), family=binomial()),
  list(resphdct ~ varhdct + Matern(1|nsloc + ewloc) + (1|pop), family=Tpoisson())),
data=byyear))

## formula_1: respld ~ varld + Matern(1 | nsloc + ewloc)
## formula_2: respfl ~ varfl + Matern(1 | nsloc + ewloc)
## formula_3: resphdct ~ varhdct + Matern(1 | nsloc + ewloc) + (1 | pop)
## Estimation of corrPars and lambda by ML (p_v approximation of logL).
## Estimation of fixed effects by ML (p_v approximation of logL).
## Estimation of lambda by 'outer' ML, maximizing logL.
## Families: 1: binomial( logit ); 2: binomial( logit ); 3: 0-truncated poisson( log )
## ----- Fixed effects (beta) -----
##           Estimate Cond. SE t-value
## (Intercept)_1    0.75962  0.61683  1.2315
## varldld03_1      1.99547  0.26999  7.3910
## varldld04_1      2.28710  0.31368  7.2912
## (Intercept)_2   -1.20246  0.61985 -1.9399
## varflfl03_2     -0.36420  0.15846 -2.2984
## varflfl04_2      0.85205  0.15630  5.4514
## (Intercept)_3   -1.24357  0.62015 -2.0053
## varhdcthdct03_3 -0.08787  0.12602 -0.6973
## varhdcthdct04_3  0.63740  0.09971  6.3925
## ----- Random effects -----
## Family: gaussian( link = identity )
## --- Correlation parameters:
##           1.nu      1.rho
## 0.1617553  0.0296269
## --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##   nsloc + e. : 1.242
##   pop : 0.01177
## # of obs: 1710; # of groups: nsloc + e., 570; pop, 7
## ----- Likelihood values -----
##           logLik
## logL      (p_v(h)): -1841.509
```

Note the large gain in log-likelihood, suggesting that these data are better fitted by models with a shared spatial random effect than by distinct spatial linear trends on each response (but we will not pursue this question here).

Now suppose that for these individuals, we had recorded some measure of individual fecundity and of subsequent growth, and wished to assess whether there is a trade-off between fecundity and growth stemming from latent individual factors. Simulated data incorporating joint random effects on **growth** and **feco** with a negative correlation `cor=-0.5` can be produced by

```
simfun <- function(nind=570L, cor=-0.5, lambda=c(0.2,0.1)) {
  u <- rnorm(2*nind)
  lam1 <- lambda[1]
  lam2 <- lambda[2]
  covmat <- matrix(c(lam1,sqrt(lam1*lam2)*cor,sqrt(lam1*lam2)*cor,lam2),ncol=2)
  L <- t(chol(covmat))
  v <- L %%% matrix(u,nrow=2)
  lfh <- data.frame(
    id=seq_len(nind),
    # rgamma() with mean=exp(1+v[2,]) and var= 0.2*mean^2:
    growth=rgamma(nind,shape=1/0.2, scale=0.2*exp(1+v[2,])),
    feco= rpois(nind, lambda = exp(1+v[1,])))
  lfh
}
set.seed(123)
lfh <- simfun()
```

To specify a model with such a correlated structure of random effect, the `mv(.)` syntax should be used, as follows: a term `(mv(1,2)|id)` declares a random effect with two correlated values for each level of the `id` grouping variable, whose correlated values each affect the successive submodels given as arguments of `mv()`. In this respect, it is conceptually similar to a random-coefficient term of the form `(submodel|id)`, if `submodel` were a factor for the two submodels specified as arguments of `mv(.)` (the latter syntax could perhaps be used, if the data were reshaped to different rows for different responses, but we try to avoid such reshaping).

Using this syntax, the fit can be achieved by

```
(troff_fit <- fitmv(submodels=list(
  list(feco ~ 1+(0+mv(1,2)|id), family=poisson()),
  list(growth ~ 1+(0+mv(1,2)|id), family=Gamma(log))),
data=lfh))

## formula_1: feco ~ 1 + (0 + mv(1, 2) | id)
```



```

## formula_2: growth ~ 1 + (0 + mv(1, 2) | id)
## Estimation of ranCoefs and phi by ML (P_v approximation of logL).
## Estimation of fixed effects by ML (P_v approximation of logL).
## Estimation of phi_2 by 'outer' ML, maximizing logL.
## Families: 1: poisson( log ); 2: Gamma( log )
## ----- Fixed effects (beta) -----
##           Estimate Cond. SE t-value
## (Intercept)_1  1.0111  0.03125  32.35
## (Intercept)_2  0.9824  0.02323  42.29
## ----- Random effects -----
## Family: gaussian( link = identity )
##           --- Random-coefficients Cov matrices:
## Group Term    Var.  Corr.
##      id .mv1  0.2056
##      id .mv2  0.08889 -0.419
## # of obs: 570; # of groups: id, 570
## ----- Residual variation -----
## * response 2 (Gamma) residual var = phi * mu^2:
## phi estimate was 0.206801
## ----- Likelihood values -----
##                               logLik
## logL      (P_v(h)): -2171.307

```

The output reads as that for a random-coefficient term. In particular, the optional `0+` in the left-hand side of the random-effect term so that (as in the quantitative-genetics application of `fitmv`) contrasts are not used: **spaMM** reports the variance estimates of the random effects each affecting each response variable, rather than the variance of an “Intercept” random variable affecting both response variables, and the variance of a “contrast” random effect added to the Intercept one for the second response variable.

Here the output looks reasonably good. But estimates of the correlation parameter may have a high variance, and designing a reliable workflow for inferring them may need more work, particularly when one of the response variables is binary.

Index

- \mathbf{X} (design matrix), [5](#)
- \mathbf{Z} (design matrix), [11](#)
- $\boldsymbol{\beta}$ (fixed-effect coefficients), [5](#)
- $\boldsymbol{\eta}$ (linear predictor), [11](#)
- λ (variance of random effects u), [5](#)
- $\boldsymbol{\mu}$ (expected response), [11](#)
- ν (smoothness of correlation), [6](#)
- $p_{\beta,v}$ (restricted likelihood approximation), [6](#)
- p_v (likelihood approximation), [6](#)
- ϕ (variance of residual error), [5](#)
- ρ (scale of correlation), [6](#)
- \mathbf{u} (random effect), [11](#)
- \mathbf{v} (random effect on linear scale), [11](#)
- inverse-Gamma, [28](#), [39](#)
- random coefficients, [30](#)
- Random-slope, *see* Random effects, random coefficients
- REML, [19](#), [39](#)
 - definition, [19](#)
 - drawbacks, [19](#)
- Response families
 - binomial, [22](#), [25](#)
 - COMPOisson, [13](#)
 - Gamma, [26](#)
 - Negative-binomial, [42](#)
 - poisson, [23](#)
- APHLs, [48](#)
- Correlation model
 - arbitrary given, [12](#)
 - Cauchy, [11](#)
 - Conditional autoregressive (CAR), [23](#)
 - IMRF, [11](#), [35](#)
 - Matérn, [5](#)
- GLM
 - canonical form, [50](#)
- HGLM, [25](#)
- Joint GLM, [29](#)
- logdet, [48](#)
- Multivariate response, [32](#)
- Prediction, [9](#)
- Random effects
 - Beta, [25](#)
 - Gamma
 - log link, [44](#)