An introduction to the `spaMM` package for mixed models

François Rousset

February 26, 2021

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 such models. This package has been first designed to fill this gap (Rousset and Ferdy, 2014). It provides likelihood-based estimates of fixed- and random-effect parameters, including spatial correlation parameters, by default using Laplace approximations when the likelihood cannot be exactly evaluated.

Initially based on the classical Matérn model for spatial correlations, `spaMM` has since been extended to fit a broader range of spatial models, models with non-spatial forms of correlation, and mixed models with non-gaussian random effects. It can handle grid-based approximations of classical geostatistical models (Lindgren et al., 2011) for which `INLA` (Lindgren and Rue, 2015) is known. It can now fit multivariate-response models (its latest major addition). It 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.

`spaMM` is continually evolving. Initial inspiration for its development came 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, some of the computations considered in such works are expensive. Hence, to make `spaMM` competitive to fit large data sets, recent versions have increasingly relied on alternative algorithms when possible, without sacrificing any of its distinctive features.

This document may serve as a tutorial for using `spaMM` and will (eventually) review the methods used therein. As a first introduction, this document does not address all aspects of inference. A series of examples is first presented in order to introduce the main functions, four basic families of responses (Gaussian, Poisson, binomial and Gamma), and to distinguish different types of models (spatial LMM, GLMM, and the wider class of HGLM including non-gaussian random effects). Later sections describe the approximations of likelihood used, introduce another response family (the Conway-Maxwell-Poisson family), multivariate-response models, and provide comparisons with alternative software.

In particular, a Gamma GLMM example will be used to show the meaning of the adjusted profile h-likelihoods (APHLs) that approximate likelihood and restricted likelihood in all models analyzed by `spaMM`, and of some other likelihood components important for the further understanding of the methods.

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

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

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

becomes

```
data("Orthodont",package = "nlme")
library(spaMM)
now <- 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 in `spaMM`. Gentle messages or warnings should attract one's attention on any other departure from deeply-established practice.

`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

$$y_i = \mathrm{fix}_i + b_i + e_i \tag{1}$$

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 ef-
fects
   pred <- sample(nb) ##  some predictor variable
   obs <- intercept+slope*pred + b +rnorm(nb,0,sqrt(resid)) ## re-
sponse
   data.frame(obs=obs,x,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 $\text{fix}_i = -1 + 0.1$`pred` for some predictor variable `pred`, random effect variance 0.5, residual error variance 0.5, and correlations of b_i which are exponentially decreasing with distance d as $\exp(-3d)$. Using standard notation for linear model, 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.

Two functions are available in `spaMM` to fit this model. We can use `fitme`, as follows:

```
HL1 <- fitme(obs~pred+Matern(1|x+y),data=d1,fixed=list(nu=0.5),method="REML")
```

or we can use `corrHLfit`, as follows:

```
HL1 <- corrHLfit(obs~pred+Matern(1|x+y),data=d1,ranFix=list(nu=0.5))
```

`corrHLfit` relies more on estimation methods (Lee et al., 2006) often involving slow steps, while `fitme` tries to avoid these steps when possible. It is recommended to use `fitme` as the default procedure.

Here the `Matern(1|x+y)` formula term means that the Matérn correlation model is fit 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 $\nu \to \infty$ 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(HL1)

## formula: obs ~ pred + Matern(1 | x + y)
## REML: Estimation of lambda, phi and corrPars by REML.
##       Estimation of fixed effects by ML.
## family: gaussian( link = identity )
##  ------------ Fixed effects (beta) ------------
##               Estimate Cond. SE t-value
## (Intercept)  -1.4482  0.49089  -2.950
## pred          0.1032  0.01219   8.464
##  --------------- Random effects ---------------
## Family: gaussian( link = identity )
##                      --- Correlation parameters:
##      1.nu     1.rho
## 0.5000000 0.8871959
##              --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    x + y  :  0.5671
##                --- Coefficients for log(lambda):
##  Group        Term Estimate Cond.SE
##  x + y (Intercept)  -0.5672  0.4886
## # of obs: 40; # of groups: x + y, 40
##  -------------- Residual variance  -------------
## Coefficients for log(phi)  ~ 1  :
##               Estimate Cond. SE
## (Intercept)  -0.4287   0.2598
## Estimate of phi=residual var:  0.6514
##  ------------- Likelihood values  -------------
##                            logLik
## p_v(h) (marginal L): -54.87225
##   p_beta,v(h) (ReL): -58.31173
```

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. A confidence interval for a fixed-effect parameter should be based on a maximum-likelihood fit, hence we first perform such a fit by using `fitme` (which performs ML fits by default), or by using `corrHLfit(.,method="ML")`, as follows:

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

and then use the `confint` function to obtain the interval:

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

## Iterative algorithm converges slowly.

## lower pred upper pred
## 0.07942996 0.12975323
```

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, method="REML")

## formula: obs ~ pred + Matern(1 | x + y)
## REML: Estimation of corrPars, lambda and phi by REML.
##       Estimation of fixed effects by ML.
## Estimation of lambda and phi by 'outer' REML, maximizing p_bv.
## family: gaussian( link = identity )
##  ------------ Fixed effects (beta) ------------
##             Estimate Cond. SE t-value
## (Intercept)  -1.4276  1.69783 -0.8408
## pred          0.1034  0.01223  8.4537
##  --------------- Random effects ---------------
## Family: gaussian( link = identity )
##                      --- Correlation parameters:
##       1.nu      1.rho
## 0.31142188 0.02485487
##              --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    x + y  :  3.242
## # of obs: 40; # of groups: x + y, 40
##  -------------- Residual variance  -------------
## phi estimate was 0.649751
##  ------------- Likelihood values  -------------
##                           logLik
## p_v(h) (marginal L): -56.22932
##   p_beta,v(h) (ReL): -58.27754
```

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, method="REML")

## formula: obs ~ pred + Matern(1 | x + y)
## REML: Estimation of corrPars, lambda and phi by REML.
##       Estimation of fixed effects by ML.
## Estimation of lambda and phi by 'outer' REML, maximizing p_bv.
## family: gaussian( link = identity )
##  ------------ Fixed effects (beta) ------------
##             Estimate Cond. SE t-value
## (Intercept)   -1.007  0.29271  -3.439
## pred           0.101  0.01088   9.276
##  --------------- Random effects ---------------
## Family: gaussian( link = identity )
##                      --- Correlation parameters:
##     1.nu    1.rho
## 16.66667 22.07269
##              --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    x + y  :  0.5745
## # of obs: 40; # of groups: x + y, 40
```

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

```
##  -------------- Residual variance  -------------
## phi estimate was 0.244756
##  ------------- Likelihood values  -------------
##                          logLik
## p_v(h) (marginal L): -48.23280
##   p_beta,v(h) (ReL): -52.59809
```

2.2 ML vs. REML

By default, `fitme` will fit all parameters jointly by ML. This default can be reversed by using `fitme(.,method="REML")` as shown in some of the previous examples.

By default `corrHLfit` will fit jointly the fixed effects by maximum likelihood (ML), and the random effect parameters by restricted ML (REML) to correct for small sample bias. For short, this is commonly denoted as an REML fit. The likelihood of the fitted model is here given by `p_v` and the restricted likelihood by `p_beta,v`. For linear mixed models, `p_v` is exactly the likelihood, and `p_beta,v` is exactly the restricted likelihood. For more general models, exact computation is not available, and `p_v` and `p_beta,v` are only approximations, discussed in later sections.

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 41.59732  1 1.121472e-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,[2] so the tests must be performed by separately fitting the two models compared. Alternativey, 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 values.

2.3 Prediction

2.3.1 Point prediction

The `predict` function returns the predicted value of the response, say $\mathbf{x}\hat{\boldsymbol{\beta}}+z\hat{v}_l$ for a new location l in space, where $\mathbf{x}$ are given values of the predictor variables; z is likewise some given value of the coefficient for the random effect $\hat{v}_l$ (being simply 1 in the above examples, and more generally obtained in the same way as the $\mathbf{Z}$ matrix); and $\hat{v}_l$ is the predicted value of the random effect(s) in the given location. For a spatial Gaussian effect this is the expected value of the Gaussian deviate given the inferred $\hat{\mathbf{v}}$'s 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.

2.3.2 Prediction variance

Prediction uncertainty can also be computed. In (spatial) mixed models, this crucially depends on the conditional variation of the random effects given the data. This variation and the uncertainty in the fixed effect coefficients jointly determine the uncertainty in the linear predictor (as measured notably by the conditional variance of the difference between its value according to the latent data-generating process, and its point prediction).

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()`

[2]A 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.

```
data(Loaloa)
lfit <- corrHLfit(cbind(npos,ntot-npos)~
           elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
           +Matern(1|longitude+latitude),method="HL(0,1)",data=Loaloa,
           family=binomial(),ranFix=list(nu=0.5,rho=2.255197,lambda=1.075))
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 preva-
lence, North Cameroon",
                               xlab="longitude",ylab="latitude"))
  }
```

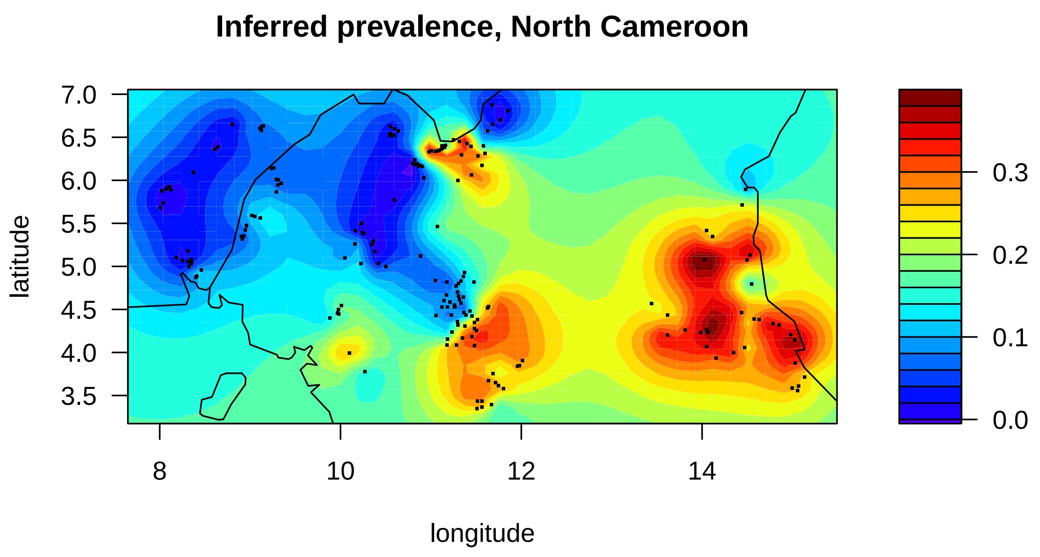

Figure 1: Plotting a map of predictions with `filled.mapMM`

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 the conditional variance of the linear predictor, excluding residual variance. `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 $\boldsymbol{\mu} = \mathrm{E}(\mathbf{y}|\mathbf{b})$ given all realized random effects $\mathbf{b}$ is written as the "linear predictor"

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

where g is the link function for the GLM response, 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).

Random-effect terms following geostatistical correlation models, conditional autoregressive model, as well as other random effects such as block effects and random-slope models, can be combined in a model.

The following geostatistical correlation models are implemented: `Matern` (as described in the above examples); `Cauchy`; grid-based approximations of classical geostatistical models (Lindgren et al., 2011), here denoted "Interpolated Markov Random Fields" (IMRFs, added in spaMM version 2.7.0; see `help("IMRF")`); and the variant introduced by Nychka et al. (2015), here

[3]Accordingly, the ith 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.

denoted “multilevel IRMFs”, where several IMRFs are controlled by common hyper-parameters.

The following autoregressive models are implemented: conditional autoregressive (CAR) models as described by an `adjacency` matrix (see example in Section 4.1), and `AR1` model.

Since version 2.6.0, it is possible to fit “autocorrelated random-coefficient” models by a syntax analogous to that of other random-coefficient terms, providing direct control of the elements of the **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 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 heteroscedasticy of the Matérn random effect). By contrast, `Matern(z|x+y)` is not defined. It could mean that a correlation structure between random intercepts and random slopes is to be combined with a Matérn correlation structure, but no way of combining them is yet defined and implemented.

3.2 Response families

3.2.1 Overview

`spaMM` fits the following response families: the base families `gaussian`, `Gamma`, `poisson`, `binomial`; the negative-binomial (using its own `negbin` implementation, which handles a shape parameter); zero-truncated variants of the Poisson and negative-binomial family (see `help(Tpoisson)` or `help(Tnegbin)` for details); and the `COMPoisson` family, which handles an underdispersion parameter (further described below).

`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 is often 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

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

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

It is a probability model of the form that can be fitted by `glm` (see `help("COMPoisson")` for examples using `glm`), and `spaMM` can also fit mixed models with this response family. Its main drawback 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 (needed for fitting by `glm`) has no explicit expression. Altogether, this implies that fitting the COMPoisson model can be relatively slow (and perhaps inaccurate) for highly overdispersed data (or, more precisely, for highly overdispersed conditional response).However, it is easier to fit models on underdipersed ($\nu_{\mathrm{CMP}} > 1$) conditional responses.

3.3 The main procedures in `spaMM`

Five functions are available for fitting random-effect models:

`fitme` and `corrHLfit` can fit linear mixed models (LMM) as just shown, and GLMMs, where the random effects are gaussian and the residual variance (i.e. conditional on the realized random effects) can be Poisson, binomial, or Gamma-distributed. They can also fit models with non-gaussian random effects such as the Beta-binomial of negative-binomial (see examples below), and models which mix gaussian (possibly spatial) and non-gaussian random effects. In this way they can fit spatial models where the residual variance (conditional on realized further random effects) is Beta-Binomial, negative binomial, etc.

These 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 lihelihood target for random-effect parameters (REML vs ML), 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 estimates of fixed effect parameters and of dispersion parameters for given correlation parameters. Only for the CAR model it also allows estimation of the correlation parameter.

`HLfit` is sufficient to fit non-spatial models, except `negbin` and `COMPoisson` response families with free overdispersion parameter, whose parameter requires the next functions to be fitted. But `HLfit` is sometimes faster than `fitme`, notably (but not necessarily) for small data sets.

`fitmv` is a multivariate extension of `fitme` (see section 4.4).

Post-fit inference functions include

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

`LRT` (or `anova`) and `fixedLRT` are 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.

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 `plot`ting he 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 moderate expected response values). The `DHARMa` package proposes more reasonable diagnostic plots, though the p-value which appears on one of them does not correspond to a generally valid goodness-of-fit test.

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

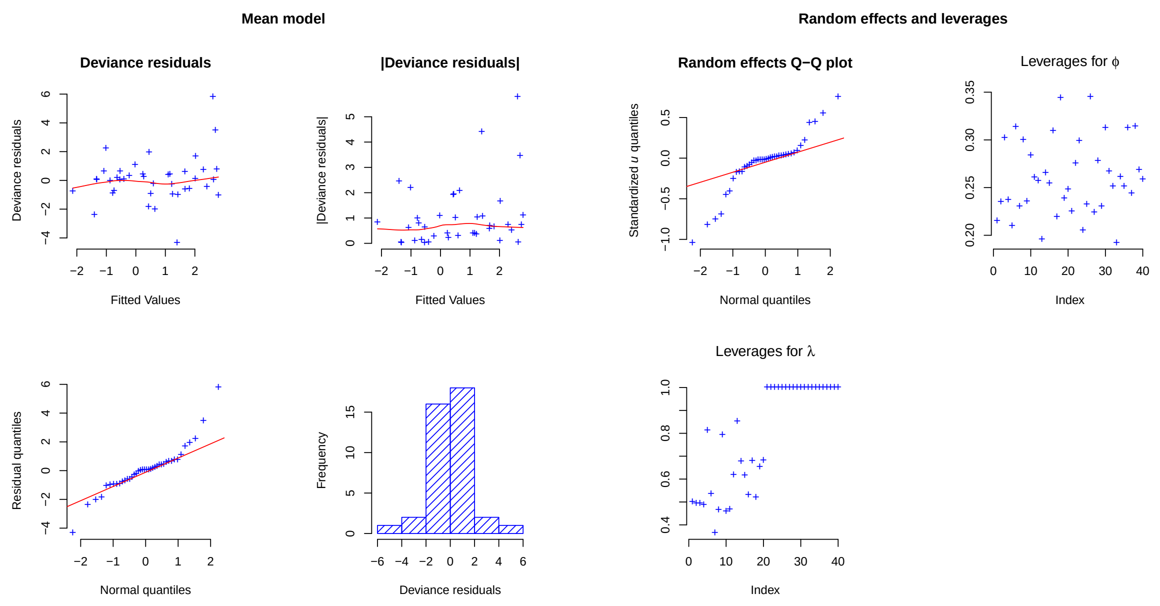

Figure 2: Diagnostic plots produced by `plot(HL1)`.

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 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 other than its average. As for other types of plots, `spaMM` offers only a crude implementation of the concept using base graphic functions, but their source code can be recycled to adapt them to your favorite graphic package.

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

```
## Non-identity link:  predVar is on linear-predictor scale.
## Warning in graphics::rug(resp, side = 2, col = colpts):
some values will be clipped
```

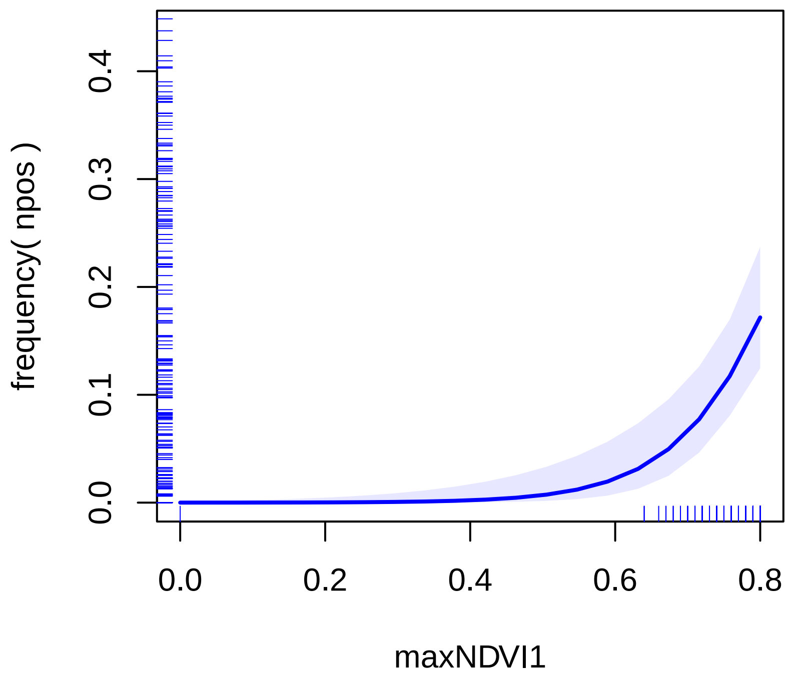

Figure 3: Partial-dependence plots produced by `plot_effects(lfit,"maxNDVI1")`.

```
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, fam-
ily=Gamma(log))
summary(glht(irisfit,mcp("Species" = "Tukey"), coef.=fixef.HLfit))
```

4 Further examples

4.1 GLMMs with autoregressive random effects

Non-Gaussian response data can be fitted by combining the `Matern` formula term together with syntax used in other procedures such as `glm` or `glmer`. For example binomial data can be fit by

```
data(Loaloa) ## parasite prevalence data in North Cameroon
binfit <- HLCor(cbind(npos,ntot-npos)~
                 1+Matern(1|longitude+latitude),data=Loaloa,
      family=binomial(),ranPars=list(nu=0.5,rho=1/0.7))
```

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

The following classical toy example (Clayton and Kaldor, 1987; Breslow and Clayton, 1993) 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

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

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_is, 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)
```

The results are very close to those of Lee and Lee (2012):

```
summary(lipfit)

## formula: cases ~ I(prop.ag/10) + adjacency(1 | gridcode) + offset(log(expec))
## Estimation of lambda by Laplace REML approximation (p_bv).
## Estimation of fixed effects by Laplace ML approximation (p_v).
## 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
## p_v(h) (marginal L): -161.5141
##   p_beta,v(h) (ReL): -163.6783
## lambda leverages numerically 1 were replaced by 1- 1e-08 (as controlled by option 'r
```

This can also be fitted using `fitme` and `corrHLfit`:[4]

[4]The results by these different functions may differ, although they are typically very similar (note the identical restricted likelihoods in this example). In particular, `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 extimation of fixed effects for given random-effect parameter

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

4.2 Beyond spatial GLMMs

Models with arbitrary fixed correlation matrix of random effects can be fitted using the `corrMatrix` argument of the `HLCor` function. This section further details various models where the correlation structure of random effects is specified by the usual `(<formula terms>|<grouping variable>)` syntax.

4.2.1 Overdispersed binomial models with crossed random effects

One can fit Binomial GLMMs using `lme4`:

```
data(salamander)
library(lme4)
glfit <- glmer(cbind(Mate,1-Mate)~TypeF+TypeM+TypeF*TypeM+(1|Female)
              +(1|Male),family=binomial(),data=salamander)
```

and this can be done with `HLfit`:

```
hlfit <- HLfit(cbind(Mate,1-Mate)~TypeF+TypeM+TypeF*TypeM+(1|Female)
              +(1|Male),family=binomial(),data=salamander,method="ML")
```

The input syntax for both procedures are exactly the same, except that `HLfit` will perform an REML fit if `method="ML"` is not specified. The results (not shown here) are also very close.

`HLfit` can also fit a Beta-binomial model.[5] As a binomial GLMM, this model assumes that the response follows a binomial distribution with expec-

estimates, and REML estimation of all random-effect parameters for given fixed-effects estimates. Further, the matrix algorithms differ, as the alternating method uses an eigen-decomposition of the adjacency matrix, while the other functions 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.

[5]see the Documentation of the `bbmle` package for a list of packages that consider the Beta-Binomial model. I have not tried them.

tation p given conditionally on a realized random effect v

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

(assuming the default logit link of the binomial GLM family). But it also assumes that

$$v = \operatorname{logit}(u) = \ln \frac{u}{1-u} \tag{6}$$

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, $p = u$ has a Beta distribution. Since v is not gaussian, this is not a GLMM, but what Lee and Nelder (1996) called a hierarchical GLM (HGLM).

We consider classical seed germination data as a toy example for Beta-binomial fits. For comparison with the results of Lee and Nelder (1996), we fit the model to these data by the method `HL(0,0)` (slightly cryptic at this step of the documentation):

```
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 Laplace REML approximation (p_bv).
## Estimation of fixed effects by h-likelihood approximation.
## family: binomial( link = logit )
##  ------------ Fixed effects (beta) ------------
##                          Estimate Cond. SE t-value
## (Intercept)              -0.46256   0.2385 -1.9397
## seed075                  -0.08003   0.3027 -0.2644
## extractCucumber           0.51480   0.3288  1.5659
## seed075: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: -42.16257
## p_v(h) (marginal L): -54.00647
##   p_beta,v(h) (ReL): -56.60452
```

The fixed effects estimates are 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).

4.2.2 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.[6] A notable feature is that it includes is 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 X1, X2 and X3 were experimentally varied. A fixed-effect model for the residual variance ("structured dispersion model") was also considered. This model can be fitted by

[6]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 given that the link between u and v is not 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 Beta binomial (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...

```
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 Laplace REML approximation (p_bv).
## Estimation of fixed effects by Laplace ML approximation (p_v).
## family: Gamma( link = log )
##  ------------ Fixed effects (beta) ------------
##             Estimate Cond. SE t-value
## (Intercept)  5.55514  0.05450 101.921
## X1           0.08376  0.02397   3.494
## X2          -0.20861  0.02397  -8.703
## X3          -0.13729  0.03786  -3.626
## I(X2^2)     -0.07641  0.02023  -3.778
## X1:X3       -0.09181  0.04019  -2.284
## X2:X3       -0.08686  0.04019  -2.161
##  --------------- Random effects ---------------
## Family: gaussian( link = identity )
##              --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    batch  :  0.02502
##                --- 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)  -2.8958   0.1384
## X3            0.1103   0.1142
## I(X3^2)       0.9468   0.1134
##  ------------- Likelihood values  -------------
##                          logLik
## p_v(h) (marginal L): -1157.609
##   p_beta,v(h) (ReL): -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 $\boldsymbol{\eta} = \ln(\boldsymbol{\mu}) = X\boldsymbol{\beta} + \mathbf{v} = \mathbf{X}\boldsymbol{\beta} + \ln(\mathbf{u})$ 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 Laplace REML approximation (p_bv).
## Estimation of fixed effects by Laplace ML approximation (p_v).
## family: Gamma( link = log )
##  ------------ Fixed effects (beta) ------------
##             Estimate Cond. SE t-value
## (Intercept)  5.56854  0.05417 102.794
## X1           0.08373  0.02396   3.494
## X2          -0.20860  0.02396  -8.706
## X3          -0.13735  0.03786  -3.628
## I(X2^2)     -0.07637  0.02022  -3.778
## X1:X3       -0.09194  0.04019  -2.287
## X2:X3       -0.08683  0.04019  -2.160
##  --------------- 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  :  0.02513
##              --- Coefficients for log(lambda):
##  Group        Term Estimate Cond.SE
##  batch (Intercept)   -3.684  0.4879
## # 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)  -2.8969   0.1384
## X3            0.1094   0.1141
## I(X3^2)       0.9479   0.1134
##  ------------- Likelihood values  -------------
##                         logLik
## p_v(h) (marginal L): -1157.523
##   p_beta,v(h) (ReL): -1175.121
```

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 Laplace REML approximation (p_bv).
## Estimation of fixed effects by ML.
## family: Gamma( link = log )
##  ------------ Fixed effects (beta) ------------
##             Estimate Cond. SE t-value
## (Intercept)  5.57570  0.03131 178.078
## 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            0.1589   0.1136
## I(X3^2)       0.7366   0.1125
##  ------------- Likelihood values  -------------
##                         logLik
## p(h)   (Likelihood): -1170.187
##   p_beta(h)   (ReL): -1189.153
```

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. In the GLM case, the `HLfit` results are quite consistent with `glm` ones (provided the correct `method` is used in the comparison) and it is easy to check analytically that the likelihood values returned by `HLfit` are more accurate than published ones.

4.3 Fitting random-slope model

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

$$\boldsymbol{\eta} = \mathbf{1}\beta_{\mathrm{I}} + \mathbf{b}_{\mathrm{I}} + \mathbf{x}_{\mathrm{S}}(\beta_{\mathrm{S}} + \mathbf{b}_{\mathrm{S}}). \tag{7}$$

The distinctive term is here $\mathbf{x}_{\mathrm{S}}\mathbf{b}_{\mathrm{S}}$ 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 $\mathbf{x}_{\mathrm{S}}$) is random, including the random effect $\mathbf{b}_\mathrm{S}$. Hence, there are two realized random effects $b_{\mathrm{I,g}}$ and $b_{\mathrm{S,g}}$ 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.

`spaMM` can fit such models by two methods: by generic function optimizers ("outer optimization": the default method through `fitme`), or by a crude extension of the iterative algorithms used to fit dispersion parameters in `HLfit`. Both methods should give consistent results, but slight differences may occur when the adjusted covariance matrix is nearly singular (see also footnote 4 for further differences between outer optimization and iterative methods). Outer optimization by `fitme` is generally more efficient for large datasets (though, for LMMs, it is still not as fast as `lmer`).

The syntax for such random effects is
`(<model term>|<grouping factor>)`,
where `<model term>` gives the explanatory variable $\mathbf{x}_\mathrm{S}$, 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)` (now handled by `spaMM`); 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 `formula`s.

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

```
HLfit(y ~X1+(X2|batch),data=wafers)

## formula: y ~ X1 + (X2 | batch)
## REML: Estimation of lambda and phi by REML.
##       Estimation of fixed effects by ML.
## 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  -------------
## Coefficients for log(phi)  ~ 1  :
##             Estimate Cond. SE
## (Intercept)    9.478   0.1035
## Estimate of phi=residual var:  13060
##  ------------- Likelihood values  -------------
##                         logLik
## p_v(h) (marginal L): -1228.721
##   p_beta,v(h) (ReL): -1222.139
## lambda leverages numerically 1 were replaced by 1- 1e-08 (as controlled by op

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

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 is what `lmer` appears to report as random effects `Variance` (and also by their `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)`.[7]

4.4 Multivariate response

Version 3.6.0 includes a new function `fitmv` to fit multivariate-response models. It is of interest mainly to fit models for different responses variables that are affected by random effects identical or correlated across the response variables (such as different phenotypes affected by unobserved genotypes). Otherwise all fits are effectively independent and all measures of uncertainty

[7]It is unclear how far such output would be useful because there is no unique representation of $\mathbf{b}$ as $\mathbf{L}\mathbf{v}$. In the present case, the covariance matrix of $\mathbf{b}$ can be represented in terms of its eigensystem, as $\mathbf{C}_\mathbf{b} = \mathbf{L}\mathbf{\Lambda}\mathbf{L}'$ where $\mathbf{L}$ contains normed eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. Thus $\mathbf{b} = \mathbf{L}\mathbf{v}$ 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 $\mathbf{P}$, $\mathbf{L}\mathbf{v}$ can be written as $(\mathbf{L}\mathbf{P})(\mathbf{P}^\top\mathbf{v})$ in terms of the permuted design matrix $\mathbf{L}\mathbf{P}$ 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, `HLfit` 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.

are also independent for the different models. Here we draw on an example taken from the aster package. It considers a life-history dataset recording survival, flowering and number of flower heads of 570 *Echinacea angustifolia* individuals other three years (2002 to 2004). In its orginal form this example has no random efects and glm fits couls as well be used (as we will see), but we will add shared random-effects.

Data preparation for analysis by aster reads as

```
library(aster)

## Le chargement a nécessité le package :  trust

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
## p(h)   (Likelihood): -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)[]
aster_pred <- predict(aout_cond, model.type="conditional")
range(c(0,sort(unique(spaMM_pred))-sort(unique(aster_pred)))

## [1] -6.704415e-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(byyear)
```

```
##        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 Laplace ML approximation (p_v).
## Estimation of fixed effects by Laplace ML approximation (p_v).
## Estimation of lambda by 'outer' ML, maximizing p_v.
## Families: 1: binomial( logit ); 2: binomial( logit ); 3: 0-truncated poisson( log )
##  ------------ Fixed effects (beta) ------------
##                 Estimate Cond. SE t-value
## (Intercept)_1    0.75962  0.61684  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
## 0.16175547 0.02962681
##            --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    nsloc + e.  :  1.242
##    pop  :  0.01177
## # of obs: 1374; # of groups: nsloc + e., 570; pop, 7
##  ------------- Likelihood values  -------------
##                          logLik
## p_v(h) (marginal L): -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

```r
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 Laplace ML approximation (p_v).
## Estimation of fixed effects by Laplace ML approximation (p_v).
## Estimation of phi_2 by 'outer' ML, maximizing p_v.
## Families: 1: poisson( log ); 2: Gamma( log )
##  ------------ Fixed effects (beta) ------------
##               Estimate Cond. SE t-value
## (Intercept)_1   1.0107  0.03132   32.27
## (Intercept)_2   0.9767  0.02264   43.13
##  --------------- Random effects ---------------
## Family: gaussian( link = identity )
##          --- Random-coefficients Cov matrices:
##  Group Term   Var.   Corr.
##     id .mv1  0.207
##     id .mv2 0.0571 -0.5638
## # of obs: 570; # of groups: id, 570
## -------------- Residual variation -------------
## * response 2 (Gamma) residual var = phi * mu^2:
## phi estimate was 0.235186
##  ------------- Likelihood values  -------------
##                           logLik
## p_v(h) (marginal L): -2172.717
```

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 has told `spaMM` to report the variance estimates of the joint values on their own rather than in a (variance of Intercept, variance of contrast of values) fashion.

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.

Future versions of `spaMM` may handle further forms of random effects cross-correlated over submodels.

5 Evaluation of the likelihood approximations

A typical fit maximizes Laplace approximations $p_v(h)$ (`p_v`) for the likelihood or $p_{\beta,v}(h)$ (`p_bv`) for the restricted likelihood. These approximations are also known as APHLs (adjusted profile h-likelihoods) in the h-likelihood literature. Based on the Gamma GLMM example, this section explains the meaning and illustrates the computation of these approximations, which are exact for LMMs. They are of the form

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

and

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

where for a given matrix $\mathbf{H}$, $\ln |\mathbf{H}|$ is the logarithm of the absolute value of its determinant (the "logdet"); and the definition of h, as well as the computation of the two distinct $\mathbf{H}$ matrices involved in $p_v(h)$ and $p_{\beta,v}(h)$, will now be explained.

For concreteness we consider fitting a simple Gamma GLMM to the data, without the residual dispersion model previously considered:

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

5.1 Conditional and h-likelihood

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

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

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 <- HLgs$fv
 nu <- 1/HLgs$phi
 clik <- with(wafers,sum(nu*log(nu*y/mui)-nu*(y)/mui-log(gamma(nu))-
log(y)))
 clik

## [1] -1180.896
```

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

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

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

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

that is

```
 hlik <- clik+with(HLgs,sum(-(ranef^2)/(2*lambda)
                             -(log(2*pi*lambda))/2))
 hlik

## [1] -1173.457
```

Here the sum is over the 11 values of v =`ranef(HLgs)`.

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

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

##      hlik
## -1173.457

logLik(HLgs, which="clik")

##      clik
## -1180.896
```

We now need some preparation to understand the computation of the logdet terms in $p_v(h)$ and $p_{\beta,v}(h)$.

5.2 GLM background for likelihood and restricted likelihood approximations

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

$$L(y; \theta, \phi) = \exp\{[y\theta - b(\theta)]/a(\phi) + c(y, \phi)\}. \tag{13}$$

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

$$\frac{\partial l}{\partial \beta_p} = \frac{\partial l}{\partial \theta}\frac{\partial \theta}{\partial \eta}\frac{\partial \eta}{\partial \beta_p} \tag{14}$$

We consider the Hessian matrix of l with respect to fixed-effect parameters, i.e. the matrix whose prth element is $\partial^2 l/\partial\beta_p\partial\beta_r$. From eq. 14, 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

$$\mathrm{E}\left[\frac{\partial^2 l}{\partial\beta_p\partial\beta_r}\right] = \mathrm{E}\left[\frac{\partial^2 l}{\partial\theta\partial\beta_r}\frac{\partial\theta}{\partial\beta_p} + \frac{\partial l}{\partial\theta}\frac{\partial^2\theta}{\partial\beta_p\partial\beta_r}\right]. \tag{15}$$

Upon sampling for given μ, $\mathrm{E}[\partial l/\partial\theta] = \mathrm{E}[y-\mu] = 0$ hence the second term is null. Therefore, we find that 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. This is useful in particular in designing iteratively reweighted least-square (IRLS) algorithms for fitting the model, and is endorsed in `spaMM` for evaluating and maximizing the Laplace approximation $p_v(h)$ for the log-likelihood, shown below.

5.3 Laplace approximation $p_v(h)$ and $p_{\beta,v}(h)$

We can now compute the elements of the matrices of which the "logdet" is required to compute p_v and $p_{\beta,v}$.[8]

[8]Usefully 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.

As in a GLM, we can write

$$\frac{\partial h}{\partial \beta_p} = \nu \sum_{ij} (y_{ij} - \mu_{ij}) w_{ij} \frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial \eta_{ij}}{\partial \beta_p} \tag{16}$$

$$\frac{\partial h}{\partial v_k} = \nu \sum_{ij} (y_{ij} - \mu_{ij}) w_{ij} \frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial \eta_{ij}}{\partial v_k} - \frac{v_k}{\lambda} \tag{17}$$

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

$$\mathbf{W} \equiv \mathrm{diag} \left(\frac{\partial \mu_{ij}}{\partial \eta_{ij}} \right)^2 / [b''(\theta)]. \tag{18}$$

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

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 $z_{ij,k} = \delta_{\mathrm{batch}(k),i}$. 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 5.2. In particular for the derivatives with respect to β_r of the different factors in (16), 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 we only consider the remaining derivative

$$\frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial (y_{ij} - \mu_{ij})}{\partial \beta_r} = - \frac{\partial \eta_{ij}}{\partial \mu_{ij}} \frac{\partial \mu_{ij}}{\partial \beta_r} = - \frac{\partial \eta_{ij}}{\partial \beta_r} = -x_{ij,r}. \tag{19}$$

Hence the nonzero elements of the expected Hessian matrix are

$$\mathrm{E} \frac{\partial^2 h}{\partial \beta_p \partial \beta_r} = -\nu \sum_{ij} x_{ij,p} \frac{\partial \eta_{ij}}{\partial \beta_p} = -\nu \sum_{ij} x_{ij,p} x_{ij,r}, \tag{20}$$

$$\mathrm{E} \frac{\partial^2 h}{\partial \beta_p \partial v_k} = -\nu \sum_{ij} x_{ij,p} z_{ij,k}, \text{ and} \tag{21}$$

$$\mathrm{E} \frac{\partial^2 h}{\partial v_k^2} = -\nu \sum_{ij} z_{ij,k}^2 - 1/\lambda. \tag{22}$$

Only the derivatives with respect to ∂v_k^2 are involved in the APHL approximation to the marginal likelihood of the fitted model, which is

$$p_v(h) = h - 0.5 \ln \left| -\frac{1}{2\pi} \frac{\partial^2 h}{\partial \mathbf{v}' \partial \mathbf{v}} \right|. \tag{23}$$

Here the Hessian matrix is diagonal, so the logdet term is simple to compute using the expected Hessian:

```
p_v <- hlik-(sum(log((nu*as.numeric(table(wafers$batch))
                     +1/HLgs$lambda)/(2*pi))))/2
p_v

## [1] -1191.096
```

but a more generally applicable algorithm is needed when the Hessian matrix is not diagonal. For example (still not aiming at efficient code), one can explicitly construct the expected Hessian matrix and compute its determinant:[9]

```
designZ <- apply(matrix(1:11,ncol=1),1,
                function(v) {as.numeric(wafers$batch==v)})
crossprods <- apply(designZ,2,function(v) {t(v)%*%designZ})
hess <- - nu[1]*crossprods -diag(rep(1/HLgs$lambda,11)) # expected Hes-
sian matrix
hlik-determinant(hess/(2*pi))$modulus[[1]]/2

## [1] -1191.096
```

A slightly more accurate approximation of the log-likelihood can be obtained by the Laplace approximation based on the observed Hessian instead of the expected one. The result is given by

```
logLik(HLgs, which="logL_Lap")

##  logL_Lap
## -1191.092
```

As previously noted this makes a difference only for non-canonical links, and is currently implemented only for the `Gamma(log)` case, for demonstration purposes.[10] However, the likelihood approximation maximized by the methods implemented in `spaMM` is $p_v(h)$ rather than this alternative approximation.

[9]`determinant(.)$modulus` directly returns the logarithm and avoids numerical overflows

[10]Here the diagonal elements of the observed Hessian are $\partial^2 h/\partial v_k^2 = -\nu \sum_{ij} z_{ij,k}^2 y_{ij}/\mu_{ij} - 1/\lambda$.

Finally, the approximation of restricted likelihood used to estimate the dispersion parameter λ depends on the Hessian for both the parameters of the fixed effects and for $\mathbf{v}$:

$$p_{\beta,v}(h) = h - 0.5 \ln \left| -\frac{1}{2\pi} \frac{\partial^2 h}{\partial(\boldsymbol{\beta}, \mathbf{v})' \partial(\boldsymbol{\beta}, \mathbf{v})} \right| . \tag{24}$$

Using brute computation rather than any simplification of the determinant:

```
designZ <- apply(matrix(1:11,ncol=1),1,
                 function(v) {as.numeric(wafers$batch==v)})
## the joint design matrix for fixed and random effects:
designXZ <- with(wafers,as.matrix(cbind(1,X1,X2,X3,X2^2,
                X1*X3,X2*X3,designZ)))
## sum-of-squares-and-products matrix
crossprods<-apply(designXZ,2,function(v) {t(v)%*%designXZ})
hess <- - nu[1]*crossprods -diag(c(rep(0,7),rep(1/HLgs$lambda,11)))
p_bv <- hlik-determinant(hess/(2*pi))$modulus[[1]]/2
p_bv

## [1] -1207.51
```

We have now recovered all four likelihood components of the Gamma GLMM fit.

6 Comparison with alternative software

In the following we compare the `spaMM` output to the output of some related packages and to a few literature results. All examples were computed with R Under development (unstable) (2021-02-24 r80033), `spaMM` 3.7.2, and the given versions of the other packages.

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

Some tricks commonly used to constrain the functions `lmer`, and `glmmPQL` (from `MASS`), to analyse spatial 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), such as `geoRglm`, can fit spatial models, and can give reasonable results, but have not been thoroughly assessed for such applications. 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.

`spaMM` includes the `Loaloa` dataset, which provides a major example (Diggle and Ribeiro, 2007; Diggle et al., 2007) of 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` (version 1.0.2.1) can fit spatial models, and can produce a fit equivalent to this call, but it is 43.9 times slower than `fitme`.

6.2 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 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.

There are many publications repeating such “faster-than” claims, but I could not find suitable comparisons, and the following ones do not support such a claim. The problem highlighted by true comparisons is that fitting the lattice model can still be slower than fitting the Matérn model because by default the lattice typically involves more vertices than the number of locations in the original data. 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.

The `loa_spde` objet 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.[11] The results of the lattice model by `spaMM` are indeed close to those of fitting the Matern 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=Loaloa)
```

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

[11]This observation can be repeated on several examples, including one given to showcase the speed on `INLA` on large data sets, provided the same lattice model is used in the fits by the methods compared.

spatial locations. 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 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.

6.3 Conditional autoregressive model: `hglm`

The vignette of the `hglm` package, version 2.2.1, presents comparisons of fitting times between `hglm` and `spaMM`, using an unspecified version of `spaMM`. `spaMM` has actually long performed much better than reported there, particularly when using the now-recommended function `fitme`. Here are cumulative timings over 10 samples for each of 6 different designs (CAR models over 50 to 500 nodes as in the original comparison; the source code is appended to this documentation) for `hglm` version 2.2.1 and `spaMM` version 3.7.2, and methods formally similar to each other (`"EQL1"` for `hglm`, `"HL(0,1)"` for `spaMM`; since these are LMMs, full REML should give the same results):

```
##      time_hglm time_HLCor time_fitme
## n50         NA       0.45       0.69
## n100      0.83       0.69       0.99
## n200      5.18       2.43       1.95
## n300     21.74      10.62       1.84
## n400     44.67      24.11       2.16
## n500     91.24      36.41       2.74
```

The initial NA reflects the fact that `hglm` fails on one of the samples (a fact which is hidden in the original report). `spaMM::HLCor` is generally faster than `hglm`, and `spaMM::fitme` is faster for the largest data sets. More elaborate comparisons could also highlight the stricter numerical criteria used by `spaMM`.

6.4 Gamma GLMM

We reconsider the previously introduced Gamma GLMM, and some variations of it. When there is no spatial correlation, so that `lme4` can now be considered, together with other packages based on some of the methods implemented in `spaMM`. `HGLMMM` (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. 22. By default, the hglm package (Rönnegård et al., 2010) returns estimates similar to those produced by HLfit with option method="EQL-". The default method in spaMM corresponds to a full Laplace approximation and has been called HL(1,1) in papers by Lee *et al.*.

6.4.1 A comparison with Lee et al.’s (2011) 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 by[12]

```
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]
```

[12]this 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.

```
## # of obs: 198; # of groups: batch, 11
##  --- Residual variation ( var = phi * mu^2 )  --
## phi was fixed.
##  ------------- Likelihood values  -------------
##                             logLik
## p_v(h) (marginal L):  -1157.663
## (Non-standard?)  ReL: -1175.251
```

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 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 HLfit 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($\genfrac{}{}{0pt}{}{0}{1}$,$\genfrac{}{}{0pt}{}{0}{1}$,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($\genfrac{}{}{0pt}{}{0}{1}$,$\genfrac{}{}{0pt}{}{0}{1}$,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.

6.4.2 Further comparisons with glmer, glmmTMB, and glmmADMB

Comparisons with glmer (from lme4 version 1.1.23) 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 (or something conceptually analogous to REML) for non-Gaussian response data. For comparison, we therefore first perform an ML fit without structured dispersion by HLfit:

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

## formula: y ~ X1 + X2 + X1 * X3 + X2 * X3 + I(X2^2) + (1 | batch)
## Estimation of lambda and phi by Laplace ML approximation (p_v).
## Estimation of fixed effects by Laplace ML approximation (p_v).
## family: Gamma( link = log )
##  ------------ Fixed effects (beta) ------------
##             Estimate Cond. SE t-value
## (Intercept)  5.61229  0.05624  99.800
## X1           0.08815  0.03262   2.702
## X2          -0.21165  0.03262  -6.488
## X3          -0.13903  0.03262  -4.262
## I(X2^2)     -0.10383  0.03264  -3.181
## X1:X3       -0.08992  0.04262  -2.110
## X2:X3       -0.08765  0.04262  -2.056
##  --------------- Random effects ---------------
## Family: gaussian( link = identity )
##            --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    batch  :  0.01916
##              --- Coefficients for log(lambda):
##  Group        Term Estimate Cond.SE
##  batch (Intercept)   -3.955  0.5159
## # 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
## p_v(h) (marginal L): -1191.025
```

`glmer` also provides fits of Gamma GLMMs:

```
library(lme4)
glmer(formula= y ~X1+X2+X1*X3+X2*X3+I(X2^2)+(1|batch),
      family=Gamma(log),data=wafers)

## Generalized linear mixed model fit by maximum likelihood (Laplace Approximati
##  Family: Gamma  ( log )
## Formula: y ~ X1 + X2 + X1 * X3 + X2 * X3 + I(X2^2) + (1 | batch)
##    Data: wafers
##       AIC       BIC    logLik  deviance  df.resid
##  2395.344  2424.939 -1188.672  2377.344       189
## Random effects:
##  Groups   Name        Std.Dev.
##  batch    (Intercept) 0.1176
##  Residual             0.3919
## Number of obs: 198, groups:  batch, 11
## Fixed Effects:
## (Intercept)           X1            X2            X3      I(X2^2)         X1:X3
##     5.60820      0.08834      -0.21153      -0.14208      -0.10351      -
0.08957     -0.08860
```

Parameter estimates slightly differ 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 `HLfit`'s approximation of the likelihood is more accurate than `glmer`'s one, and that `HLfit` 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 `HLfit` for the parameters estimates obtained with `glmer`:

```
etaLGiven <- with(wafers,5.60820+0.08834*X1-0.21153*X2
      -0.14208*X3-0.10351*X2^2-0.08957*X1*X3-0.08860*X2*X3)
wafers <- cbind(wafers,etaLGiven=etaLGiven)
HLfit( y ~(1|batch) + 0 + offset(etaLGiven),
  family=Gamma(log),data=wafers,
  ranFix=list(lambda=0.1176^2,phi=0.3919^2))

## formula: y ~ (1 | batch) + 0 + offset(etaLGiven)
## family: Gamma( link = log )
## No fixed effect
##  --------------- Random effects ---------------
## Family: gaussian( link = identity )
##           --- Variance parameters ('lambda'):
```

```
## lambda = var(u) for u ~ Gaussian;
##    batch  :  0.01383 [fixed]
## # of obs: 198; # of groups: batch, 11
##  --- Residual variation ( var = phi * mu^2 )  --
## phi was fixed to 0.153586
##  ------------- Likelihood values  -------------
##                           logLik
## p_v(h) (marginal L): -1191.271
##   p_beta,v(h) (ReL): -1191.271
```

Numerical integration also shows that the likelihood is -1191.02 for parameters estimates given by `HLfit`, which are therefore the better fit. The p_v approximation is here again very close (-1191.025), and the Laplace approximation ℓ based on the observed Hessian (as discussed in section 5.3) is $\ell = -1191.021$.

`glmmTMB` performs better that `glmer`. It reports ℓ rather than p_v, and reaches a slightly higher ℓ ($= -1191.019$). This would make sense if it maximized ℓ (whereas `spaMM` maximizes p_v).

The result with `glmmADMB`, version 0.8.3.3, is unsatisfactory:

```
library(glmmADMB)
suppressMessages(tryadmb <- try(
   glmmadmb( y ~X1+X2+X1*X3+X2*X3+I(X2^2)+(1|batch),
             family="gamma",link="log",data=wafers),silent = TRUE))

## Warning in system(cmd, intern = intern, wait = wait | intern,
show.output.on.console = wait, :  l'exécution de la commande 'C:\Windows\system3
/c glmmadmb -maxfn 500 -maxph 5 -noinit -shess' renvoie un statut
1
```

```
## [1] "Error in glmmadmb(y ~ X1 + X2 + X1 * X3 + X2 * X3 + I(X2^2) + (1 | batch
run with debug=TRUE for more information on failure mode\n"
```

6.4.3 PQL vs. `glmmPQL`

`spaMM` implements several variants of Laplace approximation, including Breslow and Clayton's (1993) PQL as also discussed by Lee and Nelder (1996).

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 <- HLfit(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 0.1649 vs 0.1508, and λ estimates are 0.02171 vs 0.01966. glmmPQL does not return likelihood values for comparison with spaMM's ones.

6.5 Negative binomial model

The standard negative binomial model can be fitted using family=negbin():

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

## formula: cases ~ I(prop.ag/10) + (1 | gridcode) + offset(log(expec))
## Estimation of fixed effects by Laplace ML approximation (p_v).
## Estimation of lambda and NB_shape by 'outer' ML, maximizing p_v.
## family: Neg.binomial(shape=2.984)( link = log )
##  ------------ Fixed effects (beta) ------------
##              Estimate Cond. SE t-value
## (Intercept)    -0.3528   0.1495  -2.359
## I(prop.ag/10)   0.7148   0.1324   5.398
##  --------------- 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
## p_v(h) (marginal L): -171.4703
```

The maximum likelihood fit of negative binomial GLMs is identical to the `glm.nb` fit (from the `MASS` package):

```
nbfit <- glm.nb(cases~I(prop.ag/10)+offset(log(scotlip$expec)),
       data=scotlip)
summary(nbfit)

##
## Call:
## glm.nb(formula = cases ~ I(prop.ag/10) + offset(log(scotlip$expec)),
##     data = scotlip, init.theta = 2.984280248, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9030  -0.8597  -0.1937   0.5310   1.7205
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.3528     0.1495  -2.359   0.0183 *
## I(prop.ag/10)   0.7148     0.1324   5.398 6.75e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(2.9843) family taken to be 1)
##
##     Null deviance: 86.474  on 55  degrees of freedom
## Residual deviance: 62.227  on 54  degrees of freedom
## AIC: 348.94
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  2.984
##           Std. Err.:  0.786
##
##  2 x log-likelihood:  -342.941
```

Alternatively, the negative binomial is the Poisson-Gamma model with $v = \ln(u)$, which can therefore be fitted with `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. However, this mixed-model representation uses a Laplace approximation for the likelihood, which yields less accurate results than `negbin` for high variance of the random effects.

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

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.

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.

Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine., *Ann. Statist.* **29**, 1189–1232.

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, Articles* **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.

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.

Rönnegård, L., Shen, X., and Alam, M. 2010. hglm: A package for fitting hierarchical generalized linear models, *R Journal* **2**, 20–27.

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.

Venables, W. N., and Ripley, B. D. 2002. “Modern applied statistics with S,” Springer-Verlag, New York, fourth edn.

Appendix: Code for comparison with `hglm`

```
## require(hglm)
## require(spaMM)
## data(ohio)
## nrepsim <- 10
## thglm <- tfitme <- tHLCor <- data.frame(n50 = rep(NA, nrepsim),
##   n100 = rep(NA, nrepsim), n200 = rep(NA, nrepsim), n300 = rep(NA,
##     nrepsim), n400 = rep(NA, nrepsim), n500 = rep(NA, nrepsim))
## n <- c(50, seq(100, 500, 100))
## rngcheck <- ("sample.kind" %in% names(formals(RNGkind)))
## if (rngcheck) suppressWarnings(RNGkind("Mersenne-Twister", "Inversion",
##   "Rounding"))
## for (k in 1:6) {
##   set.seed(911)
##   for (i in 1:nrepsim) {
##     lv <- sample(levels(ohioMedian$district), n[k])
##     idx <- which(ohioMedian$district %in% lv)
##     subdistrict <- factor(as.character(ohioMedian$district)[idx])
##     subMedian <- data.frame(MedianScore = ohioMedian$MedianScore[idx],
##       district = subdistrict)
##     subD <- ohioDistrictDistMat[levels(subdistrict), levels(subdistrict)]
##     wrapped <- try(system.time(hg1 <- hglm(fixed = MedianScore ~
##       1, random = ~1 | district, rand.family = CAR(D = subD),
##       data = subMedian, method = "EQL1")))
##     if (!inherits(wrapped, "try-error"))
##       thglm[i, k] <- wrapped[3]
##     tfitme[i, k] <- system.time(sp1 <- fitme(MedianScore ~
##       1 + adjacency(1 | district), data = subMedian, method = "HL(1,0)",
##       adjMatrix = subD))[3]
##     tHLCor[i, k] <- system.time(sp1 <- HLCor(MedianScore ~
##       1 + adjacency(1 | district), data = subMedian, HLmethod = "HL(1,0)",
##       adjMatrix = subD))[3]
##     cat(i, " ")
##   }
## }
## if (rngcheck) RNGkind("Mersenne-Twister", "Inversion", "Rejection")
## (CAR_timings <- structure(data.frame(time_hglm = colSums(thglm),
##   time_HLCor = colSums(tHLCor), time_fitme = colSums(tfitme)),
##   hglm_version = packageVersion("hglm"), spaMM_version = packageVersion("spaMM")))
```