spaMM:
an R package to fit generalized, linear and mixed models allowing for complex covariance structures

François Rousset [1] Alexandre Courtiol [2]

[1]Institut des Sciences de l'Évolution, Montpellier, France

[2]Leibniz Institute for Zoo and Wildlife Research, Berlin

With thanks to Jean-Baptiste Ferdy (University of Toulouse, France)

July 8, 2021

General features of spaMM

- First developed (Rousset & Ferdy, *Ecography*, 2014) to fit spatial Mixed Models.
- **Polyvalent**: now allows other advanced modeling (e.g. residual dispersion models; multivariate-response models; genetic correlation matrices; basic AR1 temporal model; non-gaussian random effects; COMPoisson, Zero-Truncated Poisson and ZT-negative-binomial families; Earth distance for spatial models...).
- **Simple to use**: consistent syntax across models (LMs, GLMs, LMMs, GLMMs...).
- **Robust & fast**: robust convergence for small data sets, fast fits for large data sets.

General features of spaMM

- First developed (Rousset & Ferdy, *Ecography*, 2014) to fit spatial Mixed Models.
- **Polyvalent**: now allows other advanced modeling
- **Simple to use**: consistent syntax across models (LMs, GLMs, LMMs, GLMMs...).
- **Robust & fast**: robust convergence for small data sets, fast fits for large data sets.

Methods in a nutshell:

- Laplace approximation for ML and REML (plus obscure but useful variants of penalized quasi likelihood – PQL)
- Distinct efficient matrix computations depending on sparsity of correlation matrix or random effects: sparse as in classical nested random effects, (2) dense as in spatial geostatistical, and (3) sparse inverse as in spatial autoregressive (or Markov random field) models.

A few simple examples

Model	Example of syntax in spaMM & alternatives
LM	`spaMM::fitme(y ~ X1, data = wafers)`
	`stats::lm(y ~ X1, data = wafers)`
GLM	`spaMM::fitme(y ~ X1, family = Gamma("log"), data = wafers)`
	`stats::glm(y ~ X1, family = Gamma("log"), data = wafers)`
LMM	`spaMM::fitme(y ~ X1 + (X2|batch), data = wafers)`
	`lme4::lmer(y ~ X1 + (X2|batch), data = wafers)`
GLMM	`spaMM::fitme(y ~ X1 + (X2|batch), family = Gamma("log"), data = wafers)`
	`lme4::glmer(y ~ X1 + (X2|batch), family = Gamma("log"), data = wafers)`

→ same syntax as in `stats` and `lme4` but using a single function: `fitme()`.

A few more advanced examples

Model	Example of syntax in spaMM & alternatives

"Animal"[1]	`data("DT_gryphon", package = "sommer")`
	`spaMM::fitme(BWT ~ 1 + corrMatrix(1|ID), corrMatrix = A_gryphon,`
	`data = DT_gryphon, method = "REML")`

	`prior_list <- list(G = list(G1 = list(V = matrix(p.var/2), n = 1)),`
	`R = list(V = matrix(p.var/2), n = 1))`
	`MCMCglmm::MCMCglmm(BWT ~ 1, random = ~ animal, pedigree = P_gryphon,`
	`data = Data, prior = prior_list)`

Residual-dispersion	`spaMM::fitme(y ~ 1, family = Gamma(log), resid.model = ~ X3 + I(X3^2),` `data = wafers)`
	`glmmTMB::glmmTMB(y ~ 1, family = Gamma(log), dispformula = ~ X3 + I(X3^2),` `data = wafers)`
Matérn	`spaMM::fitme(cbind(npos, ntot - npos) ~ maxNDVI1 + Matern(1|longitude + latitude),` `data = Loaloa, family = binomial())`

	`Loaloa$loc <- glmmTMB::numFactor(scale(Loaloa$latitude), scale(Loaloa$longitude))`
	`Loaloa$ID <- factor(rep(1, nrow(Loaloa)))`
	`glmmTMB::glmmTMB(cbind(npos, ntot - npos) ~ maxNDVI1 + mat(loc + 0|ID),`
	`data = Loaloa, family = binomial())`

→ same syntax from simple linear models to relatively complex models.

[1] mixed-effect model with breeding value as a random effect

Package design decisions

- We all want packages to be flexible and fast...

Package design decisions

- We all want packages to be flexible and fast...
- However, proven performance (controlled error rates...) was the first objective. The few spatial procedures available a few years ago were poor in this respect (see simulation study in Rousset & Ferdy 2014).

Package design decisions

- We all want packages to be flexible and fast...
- However, proven performance (controlled error rates...) was the first objective. The few spatial procedures available a few years ago were poor in this respect (see simulation study in Rousset & Ferdy 2014).
- "Performance" also means low non-convergence rates (e.g., for binary GLMMs; but also more robust than stats::glm() for GLMs).

Package design decisions

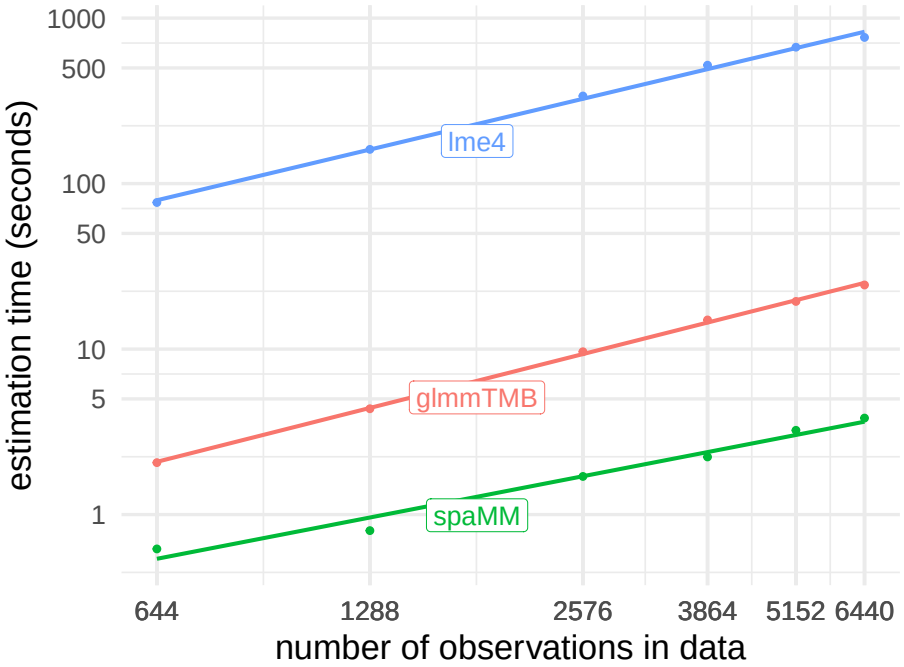
- We all want packages to be flexible and fast...
- However, proven performance (controlled error rates...) was the first objective. The few spatial procedures available a few years ago were poor in this respect (see simulation study in Rousset & Ferdy 2014).
- "Performance" also means low non-convergence rates (e.g., for binary GLMMs; but also more robust than stats::glm() for GLMs).
- Thereafter, faster procedures have been implemented without sacrificing such performance criteria.
- Still mostly R code, hence easy to extend; C++ code only for a few selected operations (and Matrix package for sparse Cholesky factorization).
- Thus, not always the fastest possible implementation; but exhibits a reasonable combination of speed, robustness and evolvability of software.

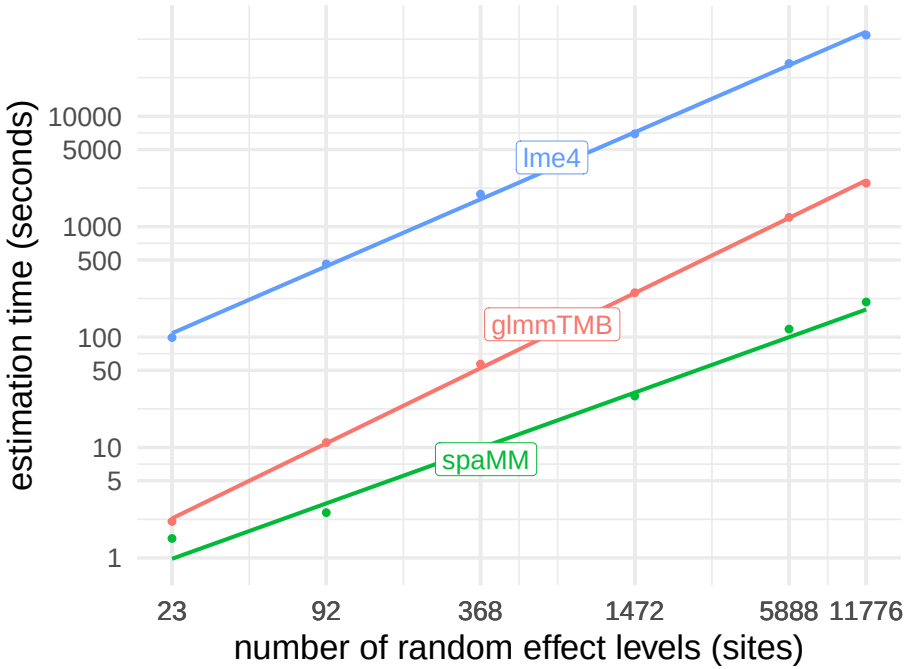
Comparisons with other packages

- Comparisons with lme4::glmer (non-spatial), INLA (spatial) and glmmTMB (both)
- Not only speed, but also expected fit results, may differ:
  - They are expected to give slightly different results for non-canonical link (e.g., Gamma(log)) because Laplace approximations slightly differ;
  - glmmTMB and spaMM should give equivalent results for GLMM with canonical link (binomial(logit), poisson(log), Gamma(inverse));
  - glmer results more often differ, even when glmmTMB and spaMM agree together.
- To limit cherry-picking, the following comparisons are based on examples used to showcase speed of glmmTMB and INLA.
- These examples are all favorable to spaMM, but don't take this too seriously (glmmTMB has some interesting features).
- Additional comparisons in the Gentle Introduction to spaMM: `https://gitlab.mbb.univ-montp2.fr/francois/spamm-ref/-/blob/master/vignettePlus/spaMMintro.pdf`

Speed: non-spatial models

Example from Brooks et al. *R Journal* (2017) Figures 1 and 2, which considers a GLMM with negative-binomial family with log link [the fitme call is **fitme**(count spp * mined + (1|grp), x, family=**negbin**())].





Note the log scale for y-axis is the plots

Comparisons for spatial models: design

- Classical Matérn correlation function for spatial random effect: can be fitted by spaMM and glmmTMB

Comparisons for spatial models: design

- Classical Matérn correlation function for spatial random effect: can be fitted by spaMM and glmmTMB
- Also Markov random field (MRF) approximation of (constrained) Matérn model: can be fitted by spaMM and INLA.

  spaMM can fit random effects with the MRF correlation model defined by some INLA functions, There are some subtleties in using these INLA functions:
  - a cutoff parameter controlling the quality of the approximation (two values are considered in next computations);
  - an alpha that corresponds to a fixed value of a smoothness parameter $\nu = 1$ of the Matérn model (alpha=2 in next computations, corresponding to $\nu = 1$)

Comparisons for spatial models: design

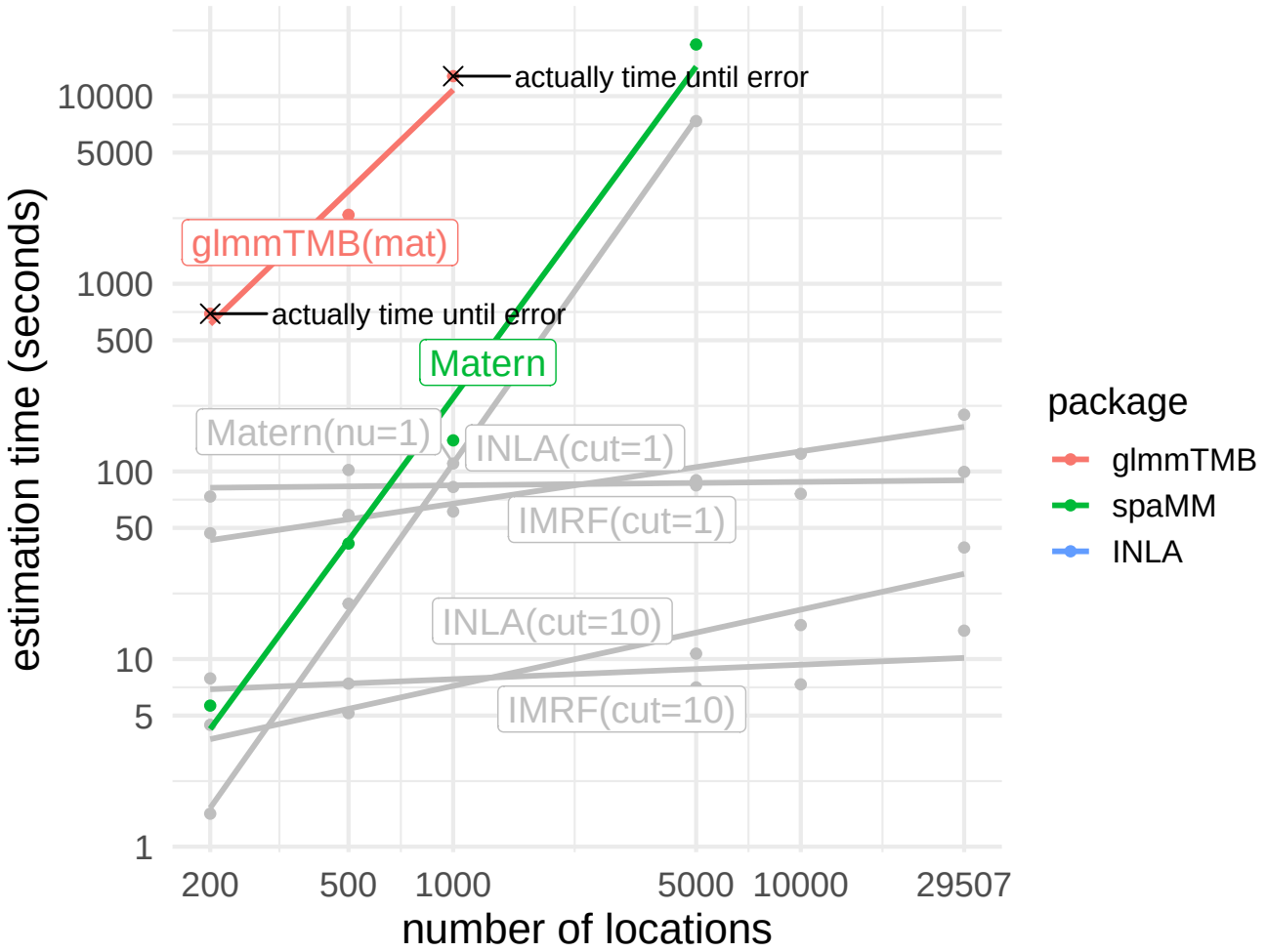
- Classical Matérn correlation function for spatial random effect: can be fitted by spaMM and glmmTMB
- Also Markov random field (MRF) approximation of (constrained) Matérn model: can be fitted by spaMM and INLA.
- Comparisons use an example provided to demonstrate INLA's efficiency (https://stat.ethz.ch/pipermail/r-sig-mixed-models/2020q3/028837.html); either the full data set of 29507 observations; or random subsets of 200, 500, 1000, 5000 and 10000 observations.

Comparisons for spatial models: design

- Classical Matérn correlation function for spatial random effect: can be fitted by spaMM and glmmTMB
- Also Markov random field (MRF) approximation of (constrained) Matérn model: can be fitted by spaMM and INLA.
- Comparisons use an example provided to demonstrate INLA's efficiency (https://stat.ethz.ch/pipermail/r-sig-mixed-models/2020q3/028837.html); either the full data set of 29507 observations; or random subsets of 200, 500, 1000, 5000 and 10000 observations.
- Simple syntax for all models and for fixing parameters:

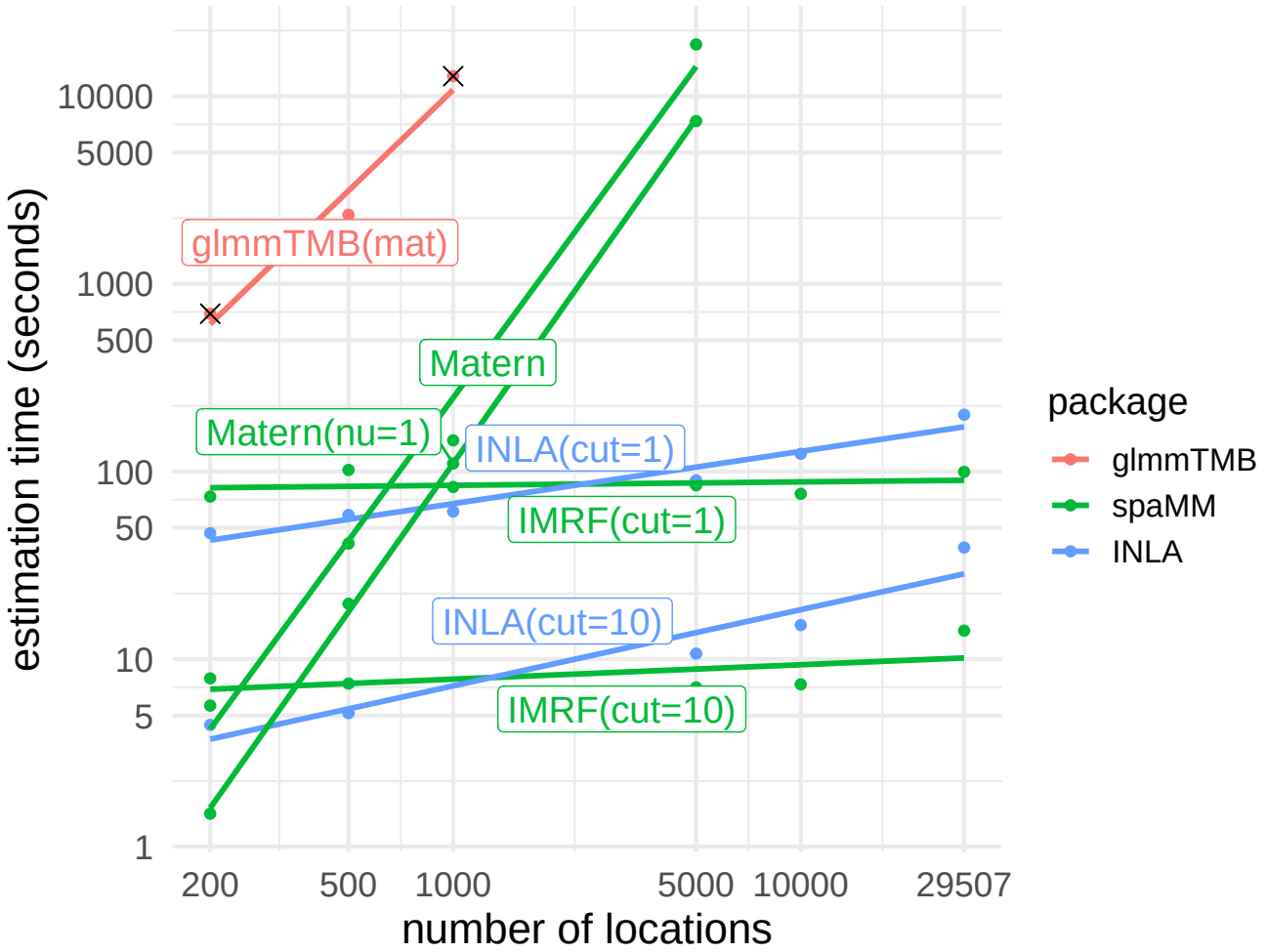
```
fitme(response ~1 + loc_x + loc_y + IMRF(1|loc_x+loc_y, model=pcmat),
      data=v, family=Gamma(log))
# where 'pcmat' correlation model provided by INLA::inla.spde2.pcmatern().
fitme(response ~1 + loc_x + loc_y + Matern(1|loc_x+loc_y),
      data=v, family=Gamma(log))
fitme(response ~1 + loc_x + loc_y + Matern(1|loc_x+loc_y),
      data=v, family=Gamma(log), fixed=list(nu=1))
```

Speed: Matérn models



spaMM is efficient, but (as is well known) computation times increase sharply with number of locations

Speed: MRF approximation of Matérn(smoothness=1)



MRF approximation is fast for large number of locations (known) and spaMM is efficient in fitting it.

Conclusions

- Polyvalent, simple to use, robust and fast;
- any model parameter can be fixed;
- Has functions for parametric bootstrap, prediction variances, marginal of conditional AIC...

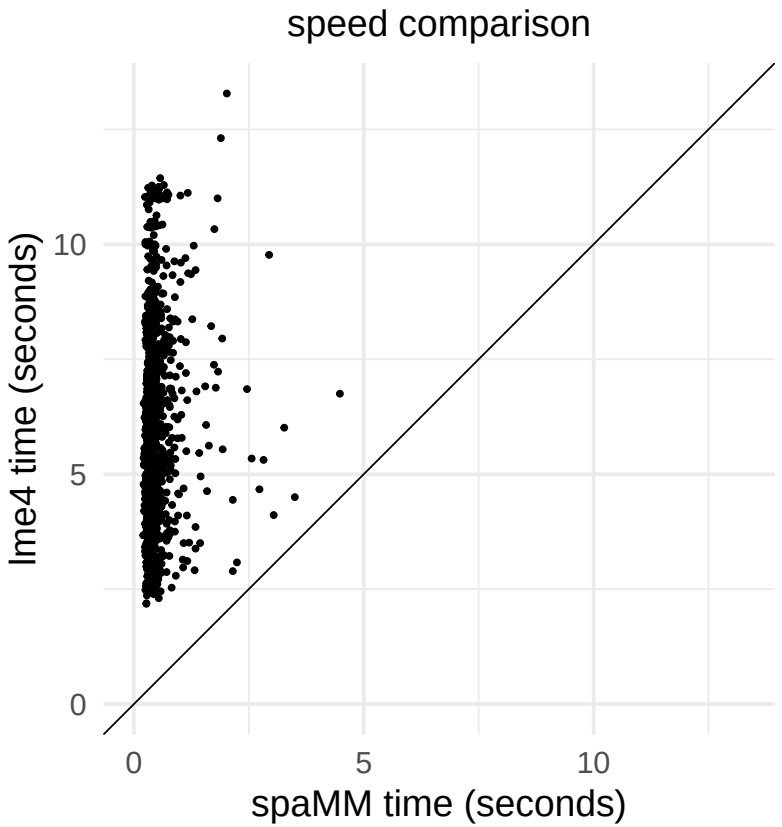
Planned developments:

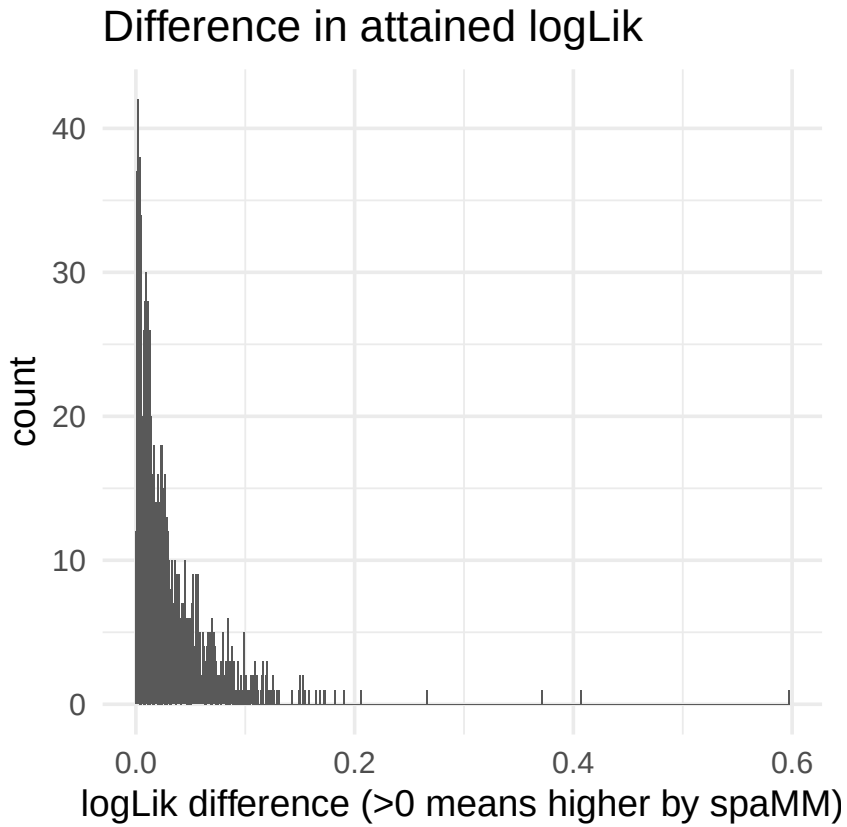
- Add parametric correlation structures for multivariate response (e.g. quantitative genetics);
- add or improve interfaces with other packages (e.g., broom.mixed, future);
- further additions potentially driven by collaborations

Comparison on non-spatial binary GLMMs

- binary GLMM, with 9 fixed-effect terms + nested random effect;
- unpublished data set of 585 observations, initially (2017) considered because fits did not converge on some bootstrap replicates;
- Assess performance on 1000 simulated bootstrap samples from the fitted model.

Binary GLMM: comparison with lme4

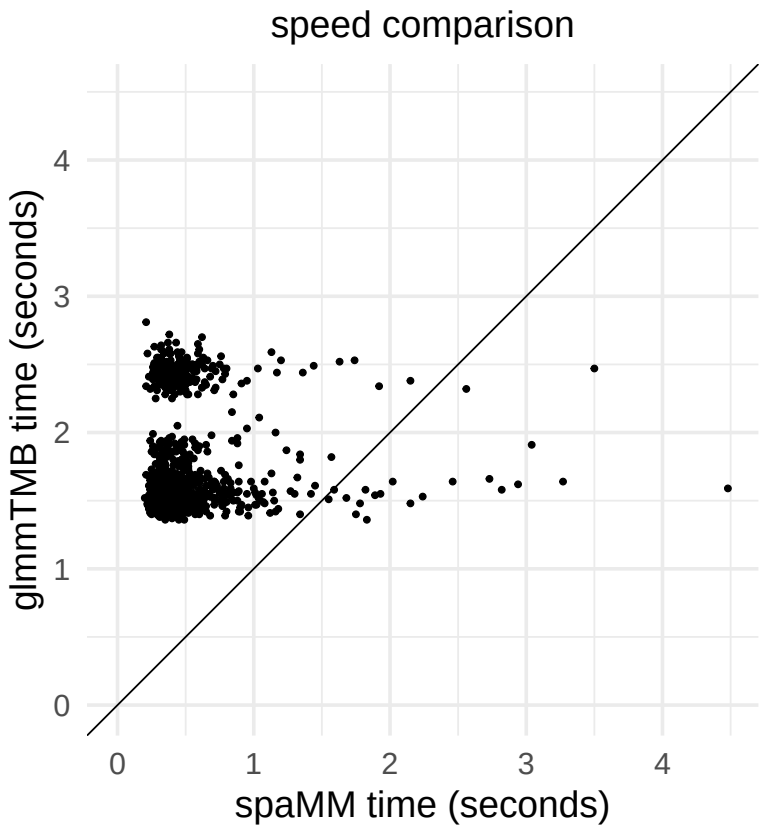


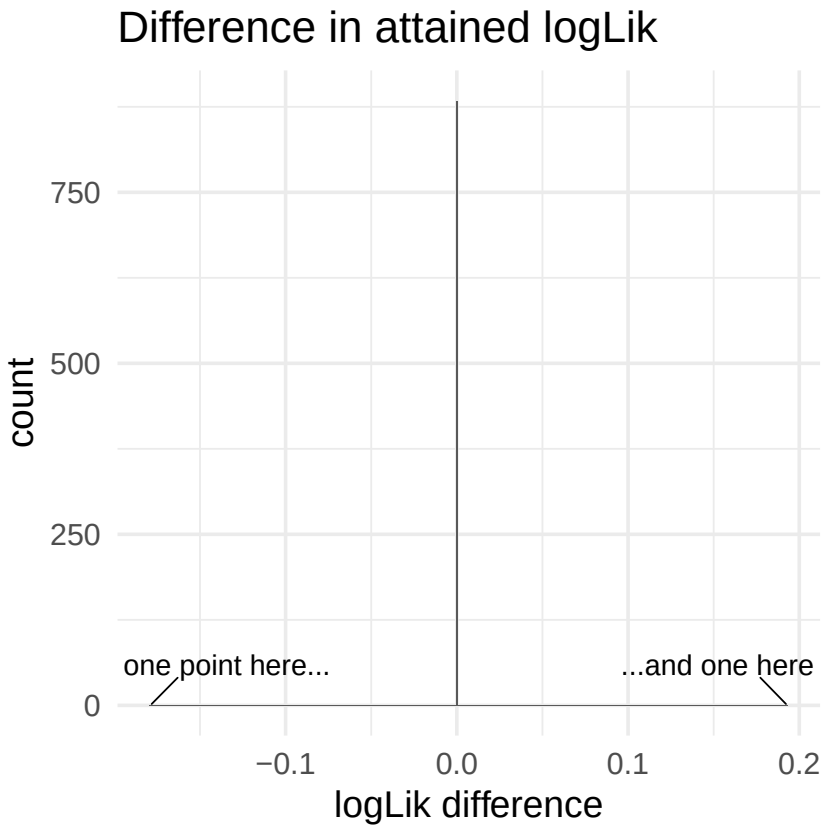


Binary GLMM: comparison with glmmTMB

Same 1000 simulated data sets: glmmTMB fails to fit 111 of them:

```
## Warning:  Removed 111 rows containing non-finite values (stat_bin).
```





```
  INLA::inla.upgrade()
```

```
## This is INLA_21.02.23 built 2021-05-08 00:36:08 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - Save 379.7Mb of storage running 'inla.prune()'
```

```
##
##  *** You already have the latest version.
```

```
  packageVersion("glmmTMB")
```

```
## [1] '1.1.1'
```

```
  packageDate("glmmTMB")
```

```
## [1] "2021-06-23"
```

```
  packageVersion("lme4")
```

```
## [1] '1.1.27.1'
```

```
  packageDate("lme4")
```

```
## [1] "2021-06-21"
```

```
  packageVersion("spaMM")
```

```
## [1] '3.8.9'
```