amer: Using lme4 to fit Generalized Additive Mixed Models

Fabian Scheipl*
LMU München

April 6, 2010

Abstract

The lme4 package uses sparse matrix technology and clever decompositions of the likelihood to fit linear, generalized, and nonlinear mixed-effects models. The amer package extends lme4's scope to include generalized *additive* mixed models (GAMM). This vignette summarizes the main ideas behind additive models and their representation in the form of a mixed model, describes the modifications to `lmer` necessary for fitting GAMMs and presents some examples with real data.

*`fabian.scheipl@stat.uni-muenchen.de`

Contents

1 Additive Models

In many applications, the assumption of a linear dependence of the response on predictor variables is inappropriate. Modelling smooth functions of an unknown shape, that is, models of the form

$$\boldsymbol{y} = \sum_{s=1}^{S} f_s(\boldsymbol{x}_s) + \boldsymbol{\varepsilon}; \quad \boldsymbol{\varepsilon} \sim N_n(0, \sigma_\varepsilon^2 \boldsymbol{I}_n)$$

where $f_s(\cdot)$ is some smooth function of a covariate $\boldsymbol{x}_i$, which can also be multidimensional (e.g. surface estimation), requires solving 3 problems not encountered in linear modelling:

1. the smooth function has to be represented somehow
2. the degree of smoothness of the function must be controllable
3. the appropriate degree of smoothness should be selected in a data-driven way

Spline smoothing addresses the first issue by assuming that $f_s(\boldsymbol{x}_s)$ can be approximated by a linear combination of d_s basis functions $B_j(\boldsymbol{x}_s)$, $j = 1, \ldots, d_s$:

$$f_s(\boldsymbol{x}_s) \approx \boldsymbol{B}_s \boldsymbol{\delta}_s; \quad \boldsymbol{B}_s = \begin{bmatrix} B_1(x_{s1}) & \ldots & B_{d_s}(x_{s1}) \\ \vdots & & \vdots \\ B_1(x_{sn}) & \ldots & B_{d_s}(x_{sn}) \end{bmatrix}$$

This obviously leads back to a linear modelling context. For ease of notation we set $S = 1$ and drop the subscript s in the following.

The second issue, controlling the roughness or "wiggliness" of the estimated function, is a variant of the bias-variance tradeoff problem: using too few basis functions may not allow the fitted curve to accurately represent the shape of the function, leading to biased estimation, while using too many will result in an overly close interpolation of the data — the estimated curve represents random noise along with the underlying structure. Penalized spline smoothing (Eilers and Marx, 1996) addresses this problem by choosing a sufficient number of knots (e.g. 10-40) to ensure the necessary flexibility of the

fit and by introducing an additional penalty term, a function of the spline coefficients $\boldsymbol{\delta}$, that quantifies the roughness of the estimated function. For a broad class of spline bases, the resulting criterion is a penalized least squares criterion,

$$\min_{\boldsymbol{\delta}} \left(\|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\delta}\|^2 + \frac{1}{\lambda}\boldsymbol{\delta}'\boldsymbol{K}\boldsymbol{\delta} \right), \tag{1}$$

where $\boldsymbol{K}$ is a penalty matrix and λ is the smoothing parameter controlling the amount of penalization, i.e. the tradeoff between fidelity to the data and complexity of the fit. The elements in $\boldsymbol{K}$ are determined by the spline basis that is used to generate $\boldsymbol{B}$ and the roughness penalty desired by the analyst (usually penalizing (local) deviations of the fitted function from a constant, linear, or a quadratic polynomial).

Example: TP-Basis
A simple example of basis functions is the truncated powers (TP) basis. A TP-Basis of degree p, with d basis functions for a covariate $\boldsymbol{x}$ and fixed knots $\kappa_1, \ldots, \kappa_{d-p}$ consists of a constant term, p global polynomial terms $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p$ and $p - d$ truncated polynomials $(\boldsymbol{x} - \kappa_i)_+^p,\ i = 1, \ldots, d - p$, where $(y)_+ = y\, I(y > 0)$:

$$\boldsymbol{B} = \left[\begin{array}{cccccccc} \boldsymbol{x}^0 & \boldsymbol{x}^1 & \ldots & \boldsymbol{x}^p & (\boldsymbol{x} - \kappa_1)_+^p & \ldots & (\boldsymbol{x} - \kappa_{d-p})_+^p \end{array} \right]$$

The penalty for the TP-Basis penalizes deviations of the fitted function from a p-degree polynomial:

$$\boldsymbol{K} = \mathrm{diag}(\boldsymbol{0}_{p+1}, \boldsymbol{1}_{d-p}).$$

The penalty term $\boldsymbol{\delta}'\boldsymbol{K}\boldsymbol{\delta}$ is simply the sum of squares of the $p - d$ coefficients for the truncated polynomials.

2 Mixed model representation of an additive model

2.1 Reparameterization: Separating penalized and unpenalized components of smooth terms

The third issue – selecting the appropriate smoothness in a data-driven way – then reduces to estimation of the smoothing parameter λ, which controls the smoothness of the estimated function. The penalized least squares problem is reformulated as a mixed model in which the smoothing parameter becomes a variance component. This is achieved by a decomposition of the spline coefficients into an unpenalized part and a penalized part:

$$\boldsymbol{\delta} = \boldsymbol{U}\boldsymbol{\beta} + \boldsymbol{P}\boldsymbol{b}$$

where $\boldsymbol{U}$, $d \times p$, is a basis of the p-dimensional nullspace of the penalization matrix $\boldsymbol{K}$ and $\boldsymbol{U}$ and $\boldsymbol{P}$ have the following properties (Kneib, 2006, ch. 5.1):

1. The matrix $[\boldsymbol{U}\boldsymbol{P}]$ has full rank to make the transformation above a one-to-one transformation. This also implies that both $\boldsymbol{U}$ and $\boldsymbol{P}$ have full column rank.
2. $\boldsymbol{U}$ and $\boldsymbol{P}$ are orthogonal, i. e. $\boldsymbol{U}\boldsymbol{P}' = \boldsymbol{0}$
3. $\boldsymbol{U}'\boldsymbol{K}\boldsymbol{U} = \boldsymbol{0}$, so that $\boldsymbol{\beta}$ is unpenalized by $\boldsymbol{K}$
4. $\boldsymbol{P}'\boldsymbol{K}\boldsymbol{P} = \boldsymbol{I}$, so that the penalty for $\boldsymbol{b}$ reduces to $\|\boldsymbol{b}\|^2$

The decomposition is not unique, but it can always be based on the spectral decomposition of $\boldsymbol{K}$. With

$$\boldsymbol{K} = [\boldsymbol{\Lambda}_+\boldsymbol{\Lambda}_0]' \begin{bmatrix} \boldsymbol{\Gamma}_+ & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} [\boldsymbol{\Lambda}_+\boldsymbol{\Lambda}_0],$$

where $\boldsymbol{\Lambda}_+$ is the matrix of the eigenvectors associated with the positive eigenvalues $\mathrm{diag}(\boldsymbol{\Gamma}_+)$, and $\boldsymbol{\Lambda}_0$ are the eigenvectors associated with the zero eigenvalues, the decomposition is

$$\begin{aligned} \boldsymbol{U} &= \boldsymbol{\Lambda}_0, \\ \boldsymbol{P} &= \boldsymbol{L}(\boldsymbol{L}'\boldsymbol{L})^{-1} \end{aligned}$$

with $\boldsymbol{L} = \boldsymbol{\Lambda}_+\boldsymbol{\Gamma}_+^{1/2}$.

Using

$$\begin{aligned} \boldsymbol{B\delta} &= \boldsymbol{B}(\boldsymbol{U\beta} + \boldsymbol{Pb}) = \boldsymbol{X_u\beta} + \boldsymbol{Z_pb} \\ \text{and} \quad \boldsymbol{\delta' K\delta} &= (\boldsymbol{U\beta} + \boldsymbol{Pb})'\boldsymbol{K}(\boldsymbol{U\beta} + \boldsymbol{Pb}) = \boldsymbol{b'b}, \end{aligned} \tag{2}$$

the penalized least squares criterion (1) can be rewritten as

$$\min_{\delta}\left(\|\boldsymbol{y} - \boldsymbol{B\delta}\|^2 + \frac{1}{\lambda}\boldsymbol{\delta' K\delta}\right) = \min_{\beta,b}\left(\|\boldsymbol{y} - \boldsymbol{X_u\beta} - \boldsymbol{Z_pb}\|^2 + \frac{1}{\lambda}\|\boldsymbol{b}\|^2\right). \tag{3}$$

For given λ, minimizing (3) over $(\boldsymbol{\beta', b'})'$ is equivalent to BLUP-estimation (Ruppert et al., 2003, ch. 4.5.3) in a linear mixed model with

$$\boldsymbol{y} = \boldsymbol{X_u\beta} + \boldsymbol{Z_pb} + \boldsymbol{\varepsilon}; \quad \boldsymbol{\varepsilon} \sim N_n(0, \sigma_\varepsilon^2\boldsymbol{I_n}); \quad \boldsymbol{b} \sim N_{d-p}(0, \sigma_\varepsilon^2\lambda\boldsymbol{I}_{d-p}),$$

which means maximizing

$$L(\boldsymbol{\beta, b}|\lambda, \sigma_\varepsilon^2) \propto \exp\left(\frac{\|\boldsymbol{y} - \boldsymbol{X_u\beta} - \boldsymbol{Z_pb}\|^2 + \frac{1}{\lambda}\|\boldsymbol{b}\|^2}{-2\sigma_\varepsilon^2}\right).$$

The reformulation of the additive model as a mixed model therefore makes it possible to estimate smoothing parameters with ML- or REML-methodology. All this is valid for non-gaussian responses as well.

2.2 Additive mixed models

This model formulation can be extended to include multiple smooth terms, other random effects and a linear predictor in the classical sense of linear regression: Just concatenate the unpenalized parts of the smooth terms to the design matrix of the fixed effects and the penalized parts of the smooth effects to the design matrix of the random effects.
For an additive mixed model

$$\boldsymbol{y} = \boldsymbol{X\beta} + \sum_{l=1}^{L}\boldsymbol{Z_l b_l} + \sum_{s=1}^{S} f(\boldsymbol{x}_s) + \varepsilon; \quad \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{\varepsilon} \end{bmatrix} \sim N_{\sum q_l + n}\left(\boldsymbol{0}, \sigma_\varepsilon^2 \begin{bmatrix} \Omega_b^{-1} & 0 \\ 0 & I_n \end{bmatrix}\right)$$

with fixed effects design $\boldsymbol{X}$, L random effects designs $\boldsymbol{Z}_l$ each with q_l parameters and random effects $\boldsymbol{b} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_L] \sim N_{\sum q_l}(\boldsymbol{0}, \sigma_\varepsilon^2 \Omega_b^{-1})$ and S smooth terms, we can write

$$\boldsymbol{y} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{Z}}\tilde{\boldsymbol{b}} + \boldsymbol{\varepsilon}$$

with concatenated design matrices

$$\tilde{\boldsymbol{X}} = [\boldsymbol{X}\boldsymbol{X}_{u,1} \ldots \boldsymbol{X}_{u,S}]\,; \quad \tilde{\boldsymbol{Z}} = [\boldsymbol{Z}_1 \ldots \boldsymbol{Z}_L \boldsymbol{Z}_{p,1} \ldots \boldsymbol{Z}_{p,S}]$$

$$\text{and} \quad \operatorname{Cov}(\tilde{\boldsymbol{b}}) = \operatorname{Cov}\left(\begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_S \end{bmatrix}\right) = \sigma_\varepsilon^2 \begin{bmatrix} \boldsymbol{\Omega}_b^{-1} & 0 & \ldots & 0 \\ 0 & \lambda_1 \boldsymbol{I}_{d_1 - p_1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & \lambda_s \boldsymbol{I}_{d_S - p_S} \end{bmatrix}$$

The parameter vector for the fixed effects is stacked in the same fashion as the one for the random effects. A minor additional complication arises from the fact that usually every matrix $\boldsymbol{X}_{u,s}$, $s = 1, \ldots, S$ for the unpenalized part of the various smooth terms will contain an intercept column. These are dropped in order to avoid a rank deficient design matrix $\tilde{\boldsymbol{X}}$. This also solves the identifiability issues common to additive models (Wood, 2006a, ch. 3.3) – we estimate a global intercept and the smooth functions etc. then only parameterize deviations from it.

2.3 Variability Estimation

The convenience functions `getF` and `plotF` to extract or plot estimated function values $\hat{f}(x)$ offer both MCMC-based or approximate frequentist variability bands. MCMC-based intervals (option `interval="MCMC"` in `getF, plotF`) are pointwise HPD-intervals. They are based on samples from lme4's `mcmcsamp` and may not be very reliable yet.[1]

The frequentist variability estimates (option `interval="RW"` in `getF, plotF`) condition on the value of the estimated variance / smoothing parameters and use the bias-adjusted covariance of $\hat{f}(x)$ derived in Ruppert et al. (2003, ch. 6.4, eq. (6.13)). See section 3.5 for an example. I plan to include bootstrap-based variability estimates in a future version.

[1] You can check the traceplots by calling
`xyplot(attr(getF(<MyModel>, interval="MCMC"),"mcmc"))`,
see section 3.5 for an example.

3 Examples

Let's fit models to some exemplary datasets to show what `amer` can do. I demonstrate how to fit simple semiparametric or additive models (3.1), how to use the `by`-option of the basis-generating function to fit group-specific smooths (3.2), how to use the `allPen`-option of the basis-generating function to fit subject- or cluster-specific smooth terms where all subject-level coefficients are penalized (i.e. the coefficients associated with $\boldsymbol{X}_u$ are treated as random effects as well) (3.3), and how to use the `varying`-option to fit varying-coefficient models (3.4). Most of the examples are adapted from Crainiceanu et al. (2005). Many additional examples on artificial data can be found in the subdirectory `tests` of the package.

3.1 Generalized Additive Model

Let's first have a look at data on wages and union membership for 534 workers described in Berndt (1991). The model assumes that the probability of union membership of worker i ($y_i = 1$ if member) depends on his or her hourly wages x_i. The smooth function is represented by a linear TP-basis:

$$\begin{aligned} P(y_i = 1) &= \text{logit}^{-1}(f(x_i)) \\ f(x_i) &= \beta_0 + \beta_1 x_i + \sum_{k=1}^{K-1} b_k (x_i - \kappa_k)_+ \\ b_k &\sim N(0, \sigma_f^2) \end{aligned}$$

We use a the default number of basis function ($K = 15$) and degree by calling the `tp`-function:

```
> data(union)
> u1 <- amer(UNION ~ tp(WAGE), family = binomial, data = union)
```

By default, `tp` uses quantile-based knot spacing to generate the basis functions. If the covariate distribution is as non-uniform as here (see figure 1), these are often a better choice than equidistant knots.

```
> print(u1, corr = F)
```

```
Generalized additive mixed model fit by the Laplace approximation
Formula: UNION ~ tp(WAGE)
   Data: union
 AIC BIC logLik deviance
 483 496   -238      477
Random effects:
 Groups Name Variance Std.Dev.
 f.WAGE tp   0.200    0.448
Number of obs: 534, groups: f.WAGE, 14

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.961      0.759   -1.27    0.206
WAGE.fx1       1.841      0.917    2.01    0.045 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note `amer`'s naming convention for the smooth function: The group name of the variance component associated with a smooth function of a covariate `x` is `f.x`, instead of the covariate name, `amer` gives the name of the basis generating function. The names of the columns in the $n \times p$ design matrix $\boldsymbol{X}_u$ for the unpenalized part of the smooth are given by `x.fx1, x.fx2` to `x.fxp`. We see that $\hat{\sigma}_f^2 \approx 0.2$. Figure 1 shows the plot produced by a call to `plotF`.

```
> plotF(u1, trans = plogis, rug = F, ylim = c(0, 0.4))
> with(union, points(WAGE, jitter(0.4 * UNION, factor = 0.15),
+     cex = 0.5))
```

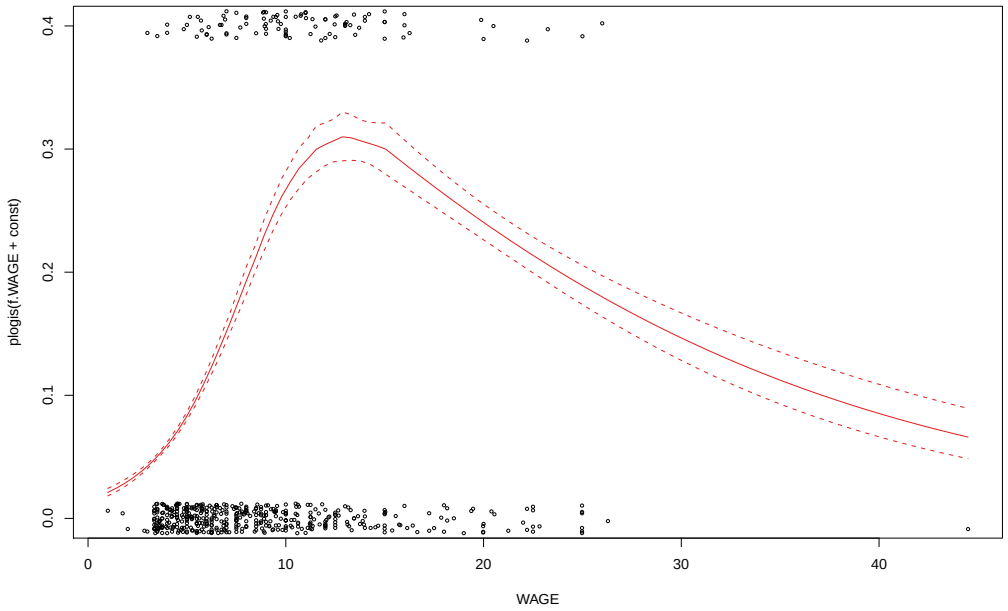

Figure 1: Fitted probability of union membership versus hourly wages, with conditional pointwise 90% CI and jittered observations.

3.2 Separate smooths for levels of a factor: Using the by-option

We use data on coronary sinus potassium concentration measurements for 36 dogs. The dogs were divided into 4 treatment groups, and the measurements for each dog were taken every two minutes from 1 to 13 minutes after occlusion (i.e. an artificially induced heart attack). The data was first published in Grizzle and Allen (1969) and previously analysed in Crainiceanu et al. (2005).

Figure 2 shows the observed concentrations for all 36 dogs split up into the treatment groups. The group-averages seem to have quite different time trends, with different degrees of nonlinearity, so we fit an additive mixed model with group-specific smooth functions $f^g_{g(i)}(t)$, $g(i) = 1, \ldots, 4$, of time and random intercepts b_0 for the different dogs:

$$
\begin{aligned}
y_{ij} &= \beta_{0g(i)} + f^g_{g(i)}(t_{ij}) + b_{0i} + \varepsilon_{ij} \\
f^g_{g(i)}(t_{ij}) &= \beta_{1g(i)} t_{ij} + \sum_{k=1}^{K-1} b^g_{g(i)k} (t_{ij} - \kappa_k)_+ \\
b^g_{g(i)k} &\sim N(0, \sigma^2_{g(i)}) \\
b_{0i} &\sim N(0, \sigma^2_{b0}) \\
\varepsilon_{ij} &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}
$$

Note that we estimate different spline coefficient variances $\sigma^2_{g(i)}$, $g(i) = 1, \ldots, 4$ for the 4 treatment groups.

Since there are only 8 unique time points for the measurements, 6 knots should be enough to model the time trends. The model is specified in `amer` using the `by`-option:

```
> d1 <- amer(y ~ -1 + group + tp(time, k = 6, by = group) +
+     (1 | dog), data = dog)

> print(d1, corr = F)

Additive mixed model fit by REML
Formula: y ~ -1 + group + tp(time, k = 6, by = group) + (1 | dog)
   Data: dog
```

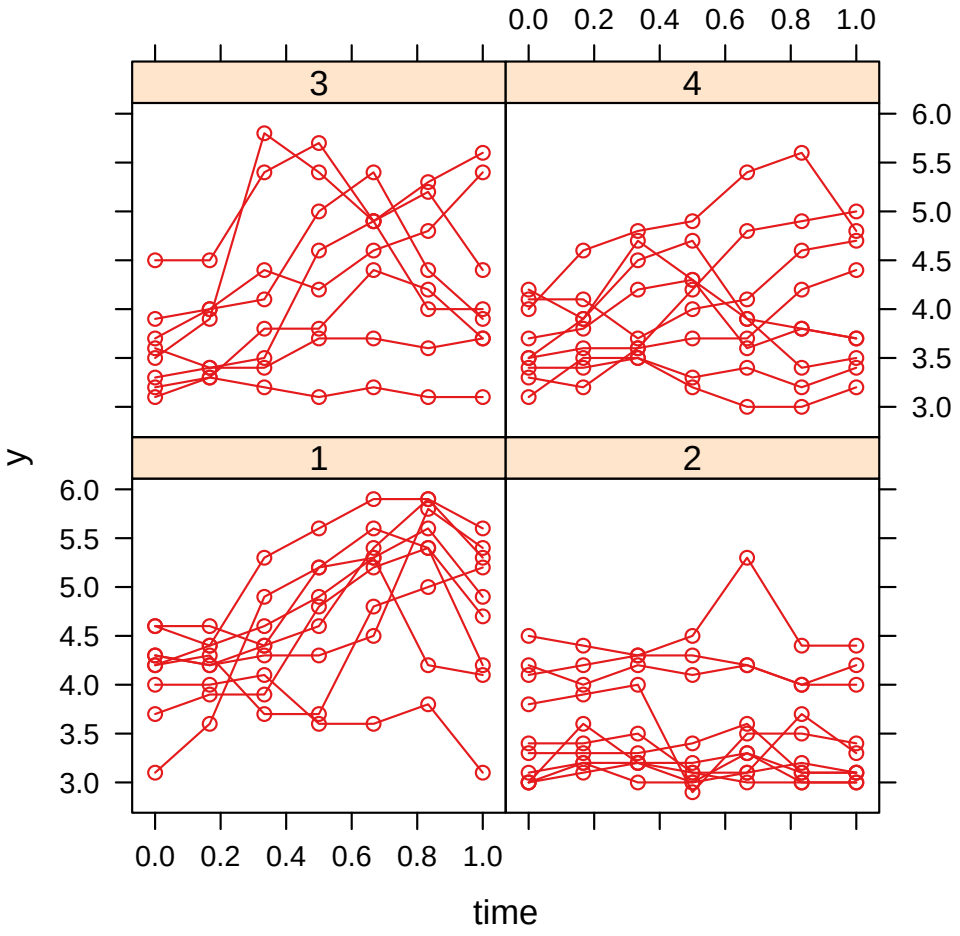

Figure 2: `dog` data: coronary sinus potassium concentrations for 36 dogs in 4 treatment groups

```
 AIC BIC logLik deviance REMLdev
 383 433   -178      341     355
Random effects:
 Groups         Name        Variance Std.Dev.
 dog            (Intercept) 0.2487   0.499
 f.time.group4 tp           0.0119   0.109
 f.time.group3 tp           0.1236   0.352
 f.time.group2 tp           0.0000   0.000
 f.time.group1 tp           0.3349   0.579
 Residual                   0.1508   0.388
Number of obs: 252, groups: dog, 36; f.time.group4, 5; f.time.group3, 5; f.time.grou

Fixed effects:
                 Estimate Std. Error t value
group1             4.4176     0.3828   11.54
group2             3.5529     0.1644   21.61
group3             4.3789     0.3350   13.07
group4             4.0325     0.2148   18.77
time.group1.fx1    0.2180     0.2808    0.78
time.group2.fx1   -0.0329     0.0465   -0.71
time.group3.fx1    0.5399     0.2395    2.25
time.group4.fx1    0.2187     0.1245    1.76
```

Note `amer`'s naming convention for smooth functions with a `by`-argument: The group name of the variance component associated with a smooth function of a covariate `x` at level `L` of the grouping factor `by` is `f.x.byL`. The names of the columns in the $n \times p$ design matrix $\boldsymbol{X}_{u,L}$ for the unpenalized part of the smooth for level `L` are given by `x.byL.fx1, x.byL.fx2` to `x.byL.fxp`.

The following code generates figure 3:

```
> layout(cbind(matrix(1, ncol = 2, nrow = 2), matrix(2:5,
+     ncol = 2, nrow = 2)))
> par(mar = c(3, 2.8, 2.8, 0.8), mgp = c(2, 1, 0))
> plotF(d1, ylim = range(dog$y), interval = "none", legend = "topleft",
+     level = 0.95, auto.layout = F, lwd = 3)
> d1.RW <- getF(d1, interval = "RW")
> for (i in 1:4) {
+     plot(0, 0, ylim = range(dog$y), xlim = c(0, 1), ylab = "y",
+         xlab = "time")
+     sub <- subset(dog, group == i)
+     lapply(split(sub, sub$dog, drop = T), function(x) lines(x$time,
+         x$y, col = "lightgrey", lty = 1, lwd = 1))
+     matlines(d1.RW[[1]][[i]][, 1], d1.RW[[1]][[i]][,
+         -1], type = "l", lty = c(1, 3, 3), col = i, lwd = 2.5)
+ }
```

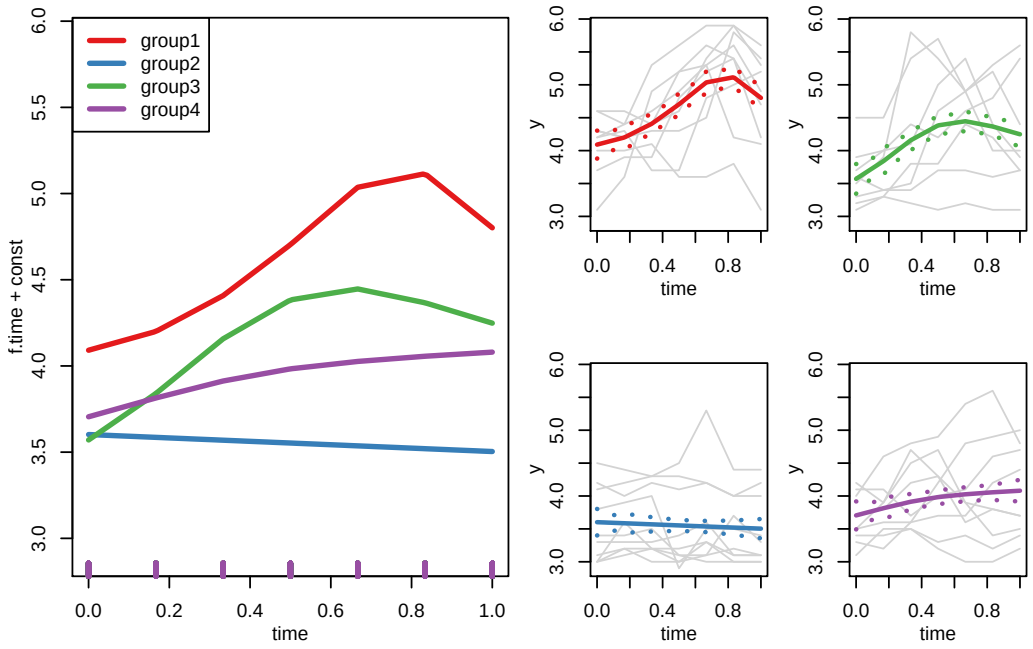

Figure 3: Left panel: Estimated groupwise smooths for the coronary sinus potassium data; Right panels: Estimated groupwise smooths with pointwise 90% CIs and observed data (grey)

3.3 Subject- or cluster-specific smooths: Using the `allPen`-option

It is also possible to allow smooth subject-specific deviations from the group-specific curves. The model is now:

$$
\begin{aligned}
y_{ij} &= \beta_{0g(i)} + f^g_{g(i)}(t_{ij}) + f^s_i(t_{ij}) + \varepsilon_{ij} \\
f^g_{g(i)}(t_{ij}) &= \beta^g_{1g(i)} t_{ij} + \sum_{k=1}^{K_g} b^g_{g(i)k}(t_{ij} - \kappa_k)_+ \\
f^s_i(t_{ij}) &= b_{0i} + b_{1i} t_{ij} + \sum_{k=1}^{K_i} b^s_{ik}(t_{ij} - \kappa_k)_+ \\
b^g_{g(i)k} &\sim N(0, \sigma^2_{g(i)}) \\
b^s_{ik} &\sim N(0, \sigma^2_{fs}) \\
(b_{0i}, b_{1i})' &\sim N_2(\mathbf{0}, \boldsymbol{D}) \\
\varepsilon_{ij} &\sim N(0, \sigma^2_\varepsilon)
\end{aligned}
$$

We still estimate separate spline coefficient variances $\sigma^2_{g(i)},\ g(i) = 1, \ldots, 4$ for the 4 treatment groups, but only one common spline coefficient variance σ^2_{fs} for all the subject-specific smooth functions $f^s_i(t)$. We assume an unstructured covariance matrix $\boldsymbol{D}$ for the subject-specific random intercepts and slopes (b_{0i}, b_{1i}).

The model is specified in `amer` by using the `by`-option in combination with `allPen = TRUE`:

```
> d2 <- amer(y ~ -1 + group + tp(time, k = 6, by = dog,
+     allPen = T) + tp(time, k = 6, by = group), data = dog)

> print(d2, corr = F)

Additive mixed model fit by REML
Formula: y ~ -1 + group + tp(time, k = 6, by = dog, allPen = T) + tp(time,      k =
   Data: dog
 AIC BIC logLik deviance REMLdev
 348 408   -157      298     314
Random effects:
 Groups        Name         Variance Std.Dev. Corr
```

```
 f.time.dog    tp           0.03503  0.1872
 u.time.dog    (Intercept)  0.25058  0.5006
               time.dog.fx1 0.00453  0.0673   1.000
 f.time.group4 tp           0.01236  0.1112
 f.time.group3 tp           0.12478  0.3532
 f.time.group2 tp           0.00000  0.0000
 f.time.group1 tp           0.37166  0.6096
 Residual                   0.09315  0.3052
Number of obs: 252, groups: f.time.dog, 180; u.time.dog, 36; f.time.group4, 5; f.tim

Fixed effects:
                Estimate Std. Error t value
group1           4.38151    0.34252   12.79
group2           3.57606    0.17826   20.06
group3           4.33032    0.30996   13.97
group4           4.07611    0.21710   18.77
time.group1.fx1  0.18889    0.24172    0.78
time.group2.fx1 -0.00977    0.07936   -0.12
time.group3.fx1  0.50846    0.21176    2.40
time.group4.fx1  0.26291    0.12545    2.10
```

By specifying `allPen = TRUE`, a random intercept for the `by`-variable is automatically included in the model. Also note `amer`'s naming convention for smooth functions of a covariate `x` with a `by`-argument and `allPen = TRUE`: For the random effects associated with $\boldsymbol{X}_u$, the group name of the variance component is `u.x.by`. The factor `u.x.by` is of course the same as `by`, the renaming is done for technical reasons.

Especially for spline bases with a higher dimensional nullspace of the penalty it may not be feasible or desirable to estimate an unstructured covariance matrix $\boldsymbol{D}$. By setting the `diag`-option to `TRUE` in the specification of a smooth term with `allPen = TRUE`, we can enforce uncorrelated random effects for the coefficients associated with $\boldsymbol{X}_u$:

```
> d3 <- amer(y ~ -1 + group + tp(time, k = 6, by = dog,
+     allPen = T, diag = T) + tp(time, k = 6, by = group),
+     data = dog)

> print(d3, corr = F)

Additive mixed model fit by REML
Formula: y ~ -1 + group + tp(time, k = 6, by = dog, allPen = T, diag = T) +      tp(
```

```
  Data: dog
 AIC BIC logLik deviance REMLdev
 348 404   -158      300     316
Random effects:
 Groups        Name         Variance Std.Dev.
 f.time.dog    tp           0.0353   0.188
 u.time.dog    time.dog.fx1 0.0000   0.000
 u.time.dog    (Intercept)  0.1972   0.444
 f.time.group4 tp           0.0122   0.111
 f.time.group3 tp           0.1245   0.353
 f.time.group2 tp           0.0000   0.000
 f.time.group1 tp           0.3714   0.609
 Residual                   0.0944   0.307
Number of obs: 252, groups: f.time.dog, 180; u.time.dog, 36; f.time.group4, 5; f.tim

Fixed effects:
               Estimate Std. Error t value
group1          4.38218    0.33499   13.08
group2          3.57596    0.16279   21.97
group3          4.33194    0.29969   14.45
group4          4.07510    0.20304   20.07
time.group1.fx1 0.18943    0.24185    0.78
time.group2.fx1 -0.00987   0.07682   -0.13
time.group3.fx1 0.50957    0.21118    2.41
time.group4.fx1 0.26190    0.12360    2.12
```

3.4 Varying coefficient models: Using the `varying`-option

Another class of models that can be fitted with amer are varying coefficient models. They are used to model smoothly varying regression coefficients, i.e. models in which the effect of a covariate z varies smoothly over the range of another covariate x ($\cdot$ denotes elementwise multiplication of the columns):

$$\begin{aligned} \boldsymbol{y} &= \beta(\boldsymbol{x}) \cdot \boldsymbol{z} + \boldsymbol{\varepsilon} \\ \beta(\boldsymbol{x}) &= f(\boldsymbol{x}) \approx \boldsymbol{X_u}\boldsymbol{\beta} + \boldsymbol{Z_p}\boldsymbol{b} \\ \Rightarrow \beta(\boldsymbol{x}) \cdot \boldsymbol{z} &\approx (\boldsymbol{X_u} \cdot \boldsymbol{z})\boldsymbol{\beta} + (\boldsymbol{Z_p} \cdot \boldsymbol{z})\boldsymbol{b} \end{aligned}$$

This class of models can be fitted by simply scaling the design matrices for the spline of the effect-modifying covariate x (i.e. the varying coefficient) with the values of the covariate z. A slight complication arises: for all other classes of models, we drop the intercept column in $\boldsymbol{X}_u$ so that the model is identifiable. That is unnecessary in this case, so the design matrix $(\boldsymbol{X_u} \cdot \boldsymbol{z})$ has $\mathbf{1} \cdot \boldsymbol{z} = \boldsymbol{z}$ as its first column.

Let's look at lattice's `ethanol` data set as an example: Ethanol fuel was burned in a single-cylinder engine. For various settings of the engine compression (`C`) and the equivalence ratio (`E`, a measure of the richness of the air and ethanol fuel mixture), the emissions of nitrogen oxides (`NOx`) were recorded. We assume that, for a given equivalence ratio `E`, the relationship between compression and emissions is linear, but with different intercepts and slopes for different values of `E` (see figure 4).

The model we want to fit is

$$\begin{aligned} \texttt{NOx}_i &= f_1(\texttt{E}_i) + f_2(\texttt{E}_i)\texttt{C}_i + \varepsilon_i \\ f_1(\texttt{E}) &= \beta_0 + \boldsymbol{X}_{\boldsymbol{u}}^{(\boldsymbol{E})}\boldsymbol{\beta}^{(\boldsymbol{E})} + \boldsymbol{Z}_{\boldsymbol{p}}^{(\boldsymbol{E})}\boldsymbol{b}^{(\boldsymbol{E})} \\ f_2(\texttt{E})\texttt{C} &= \boldsymbol{X}_{\boldsymbol{u}}^{(\boldsymbol{EC})}\boldsymbol{\beta}^{(\boldsymbol{EC})} + \boldsymbol{Z}_{\boldsymbol{p}}^{(\boldsymbol{EC})}\boldsymbol{b}^{(\boldsymbol{EC})}, \end{aligned}$$

with the usual distributional assumptions about $\boldsymbol{b}^{(\boldsymbol{E})}, \boldsymbol{b}^{(\boldsymbol{EC})}$ and $\boldsymbol{\varepsilon}$. The command to fit this model in `amer` is simply

```
> e1 <- amer(NOx ~ tp(E, k = 20) + tp(E, k = 20, varying = C),
+     data = ethanol)

> print(e1, corr = F)
```

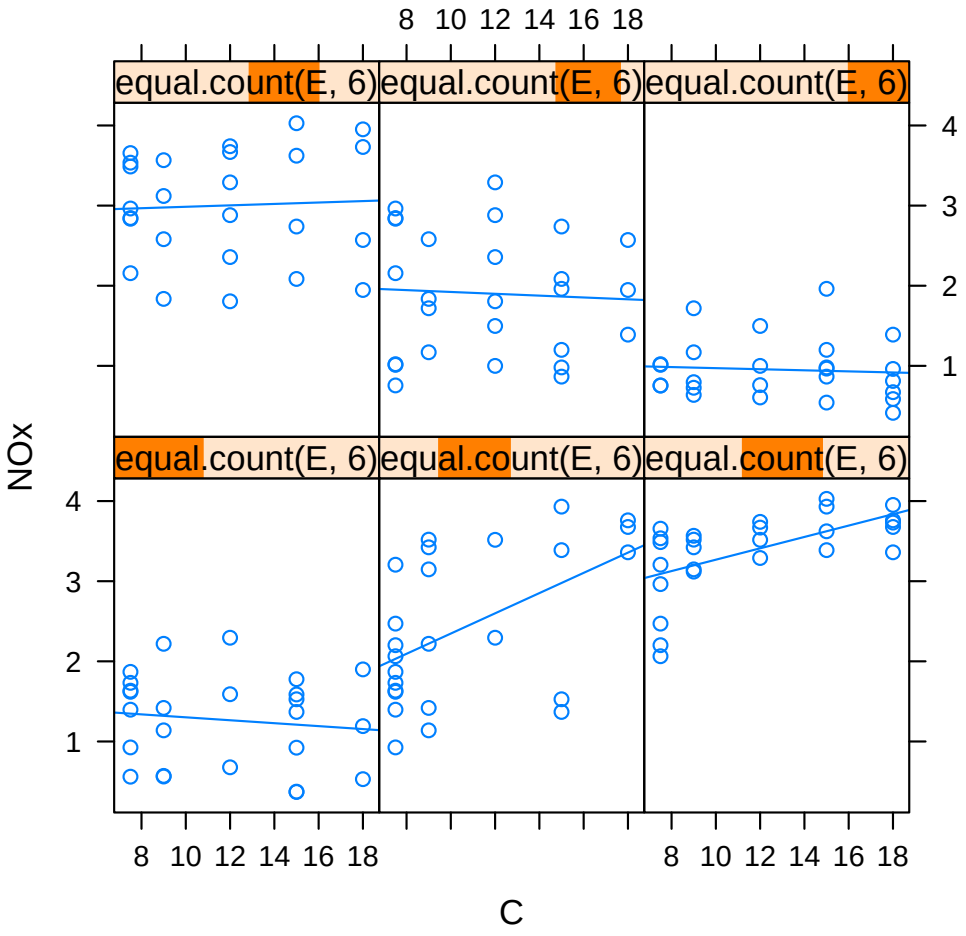

Figure 4: Emissions of nitrogen oxides `NOx` for various engine compression values `C`, split up according to equivalence ratio `E`. Lines are linear regression estimates for the subgroups.

```
Additive mixed model fit by REML
Formula: NOx ~ tp(E, k = 20) + tp(E, k = 20, varying = C)
   Data: ethanol
  AIC  BIC logLik deviance REMLdev
 15.1 32.4 -0.554      -15    1.11
Random effects:
 Groups   Name Variance Std.Dev.
 f.E      tp   1.26248  1.1236
 f.EXC    tp   0.00103  0.0321
 Residual      0.02936  0.1714
Number of obs: 88, groups: f.E, 19; f.EXC, 19

Fixed effects:
            Estimate Std. Error t value
(Intercept)   2.2834     1.0528    2.17
E.fx1         1.6808     0.7528    2.23
EXC.fx1       0.1389     0.0540    2.57
EXC.fx2       0.0317     0.0394    0.80
```

The fit is plotted in figure 5. By default, the value of the varying coefficient function (right column) is evaluated for a covariate value of $z = 1$, so the plot for `f.EXC` can be interpreted directly as $\beta(\mathtt{E})$.

Note `amer`'s naming convention for varying coefficient models: For an effect-modifying covariate `x` and an effect-causing covariate `z`, the function name is given as `f.xXz`, the unpenalized effects are named `xXz.fx1` to `xXz.fxp`. The first unpenalized effect corresponds to the conventional regression coefficient for `z`, since $\boldsymbol{X_u}$ has $\boldsymbol{z}$ as its first column.

3.5 Variability Bands: Using the `RW`- and `MCMC`-options

By default, `amer`'s `plotF` computes approximate pointwise intervals for the smooth functions based on a bias-adjusted (Ruppert et al., 2003, ch. 6.4) approximation of $\mathrm{Cov}((\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{b}} - \boldsymbol{b})|\hat{\sigma}_b^2, \hat{\sigma_\varepsilon^2})$ (see 2.3). These may underestimate the true variability since they ignore the uncertainty in the estimated variance parameters.

Alternatively, the results from `lme4`'s `mcmcsamp` can be used to construct MCMC-based variability bands. Figure 5 compares the frequentist, bias-adjusted variability bands (see 2.3) for the estimated function values with pointwise HPD-Intervals based on 1000 draws from `mcmcsamp` for the `ethanol` data. We expect the latter to be wider since they take into account the variability of the estimated variances, while the former are conditioned on the estimated variances.

Since `mcmcsamp` may not always work as expected, it is strongly recommended to examine the returned MCMC-samples. They are available as the `mcmc`-attribute of the value returned by `getF` or `plotF`. A quick visual inspection of the sampling paths can be done via `xyplot`, see figure 6. The MCMC iterations in this case show strange spikes after about 700 iterations with humongous values drawn for the relative standard deviations `e1@ST` of the spline coefficients. The marginal posterior densities for the variance components are concentrated on values magnitudes larger than the REML estimates found by the optimizer, with ridiculously long upper tails (which lead to quite erratic sampling behaviour of the spline coefficients $\boldsymbol{b}$ and consequently very broad HPD-Intervals for $\hat{f}(\boldsymbol{x})$).

```
> par(mfrow = c(2, 2), mar = c(3, 2.8, 2.8, 0.8), mgp = c(2,
+     1, 0))
> e1.RW <- plotF(e1, addConst = c(T, F), level = 0.95,
+     auto.layout = F)
> set.seed(12345)
> e1.MCMC <- plotF(e1, addConst = c(T, F), int = "MCMC",
+     sims = 1000, level = 0.95, auto.layout = F)

starting 1000 MCMC iterations for posterior intervals:
                ... done.
```

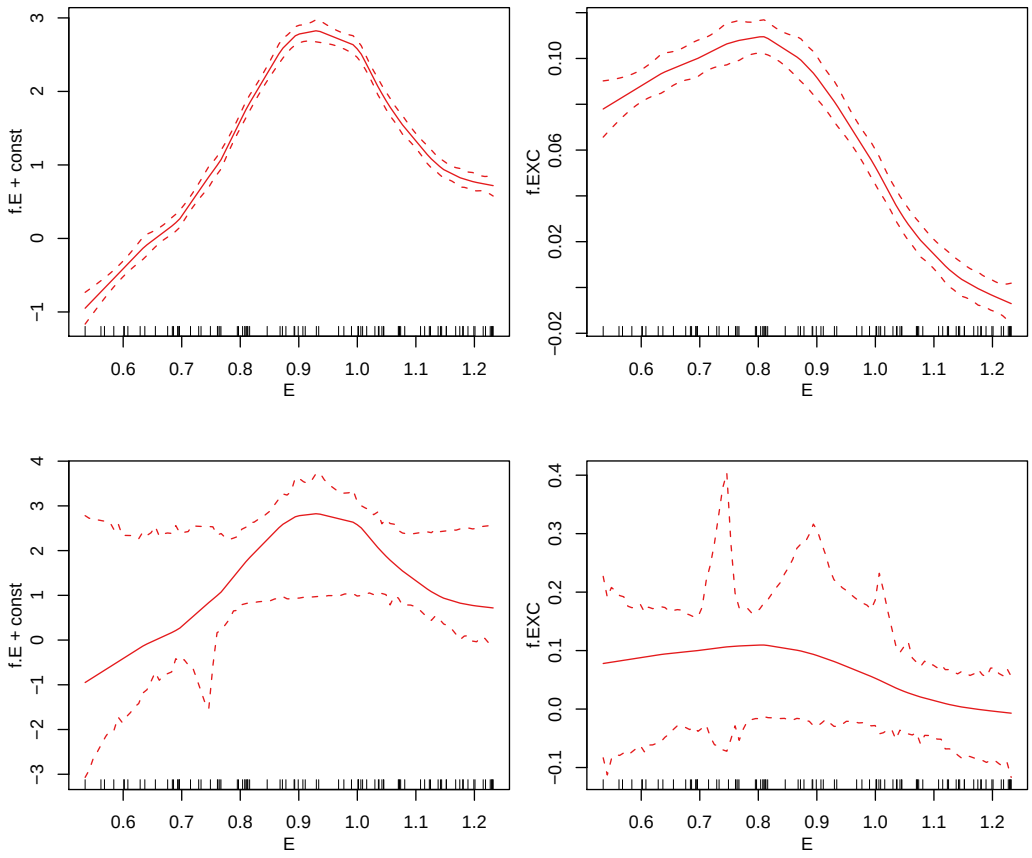

Figure 5: Fits and 95% CIs/HPD-Intervals for the `ethanol` data. Upper row: pointwise frequentist variability bands conditional on the estimated variances. Lower row: pointwise HPD-Intervals based on 1000 draws (see text) from `mcmcsamp`. Left column: Effect of equivalence ratio `E`. Right column: regression coefficient for compression `C` varying over `E`.

```
> e1.MCMCData <- as.data.frame(attr(e1.MCMC, "mcmc"))
> data.frame(c(fixef(e1), e1@ST, lme4:::sigma(e1)))

   X.Intercept. E.fx1 EXC.fx1 EXC.fx2   tp  tp.1 sigmaREML
tp         2.28  1.68   0.139  0.0317 6.56 0.187     0.171

> apply(e1.MCMCData, 2, quantile, probs = c(0.1, 0.25,
+     0.5, 0.75, 0.9), na.rm = T)

     (Intercept)   E.fx1  EXC.fx1  EXC.fx2   ST1      ST2 sigma
10%        0.195 -0.2168 -0.01897 -0.05728   0.0      0.0 0.168
25%        1.170  0.0763  0.00539 -0.03471   0.0      0.0 0.207
50%        1.914  0.2643  0.06521  0.00906   0.0      0.8 0.273
75%        3.263  2.4638  0.14134  0.08058  31.7     89.2 0.663
90%        5.048  3.7896  0.29875  0.15520 169.3 57125.2 0.992


> print(xyplot(attr(e1.MCMC, "mcmc")))
```

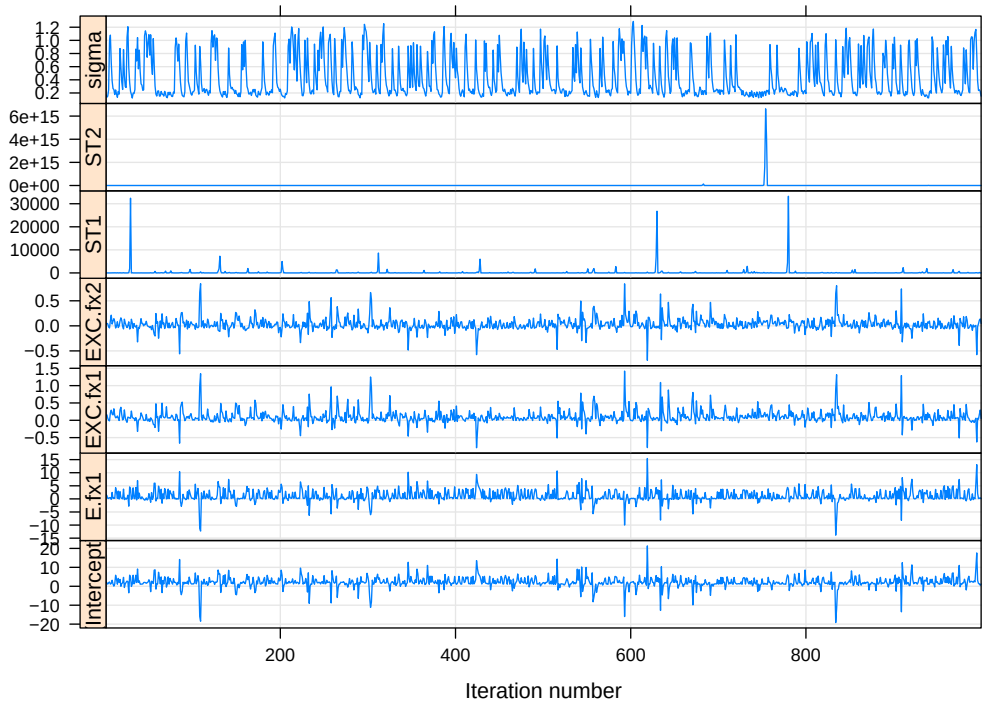

Figure 6: Traceplot of the MCMC samples for model `e1`. Note the huge values for `ST`.

4 Implementation

4.1 What's the point?

There is already a well-tested, well documented and versatile package mgcv (Wood, 2006b) that fits generalized additive mixed models in R, so why bother with yet another one?

- mgcv's `gamm` uses the less stable and slower nlme-implementation of linear mixed models, while amer relies on the more stable algorithm used in lme4 (Bates and Maechler, 2009b) with its very fast sparse Matrix (Bates and Maechler, 2009a) magic. Also, specifying random effects terms for `amer/lmer` does not use `gamm/nlme`'s cumbersome list notation.
- mgcv's `gamm` fits non-gaussian responses by calling MASS's `glmmPQL`. The PQL-approach for fitting generalized linear mixed models (GLMM) is severely biased and often unstable. amer relies on the more precise Laplace approximation implemented in lme4 for fitting GLMMs.
- asreml also offers additive models, but is limited to gaussian responses (and isn't free or open-source)

The `/tests` file `mgcvTests.R` does some comparisons between `gamm` and `amer`. The drawbacks of using amer instead of mgcv's `gamm` are that, as yet, it's not possible to include serial and/or spatial correlation structures or variance functions for the residuals or specify covariance structures of the random effects that aren't either diagonal or unstructured (this will remain an issue as long as lme4 doesn't have that capability). Also, multidimensional smooths are not yet implemented, but will be included in a future version.

4.2 Making `lmer` fit GAMMs

In its most current version (`0.999375-33`, at the time of writing), lme4 fits mixed models

- for hierachical data structures (i.e. grouped data) and

- only admits either diagonal or unstructured covariances of the non-scalar random effects for every level of grouping.

In an additive mixed model, data are not grouped (the smoothing introduces dependence between all the observation in the data) and, if the reparameterization from $\boldsymbol{B\delta}$ to $\boldsymbol{X_u\beta + Z_pb}$ is not done, the precision matrix $\boldsymbol{K}/\lambda$ for the penalized coefficients is in general not diagonal (and certainly not unstructured) and not of full rank, so that an implementation of GAMMs based on the unreparameterized representation is not possible without changing the underlying C-code of lme4.

Instead of making these changes to the underlying C, `amer` tricks `lmer` into fitting additive models by setting up an unfit model object with the structure of random and fixed effect design matrices necessary for the mixed model representation (2) of the additive model, and then overwriting the (precursor of) the `Zt`-slots with the penalized parts $\boldsymbol{Z}_p$ of the reparameterized spline bases. More precisely, the model object is set up by going through the following steps for each smooth function:

1. Generate $\boldsymbol{X}_u$ and $\boldsymbol{Z}_p$ according to the basis generating function (see section 4.4) given for the smooth term,
2. replace the smooth term in the original model formula with fixed effect terms for the columns in $\boldsymbol{X}_u$ and a random intercept term for an artificial grouping factor that has as many levels as $\boldsymbol{Z}_p$ has columns,
3. add the fixed effects in $\boldsymbol{X}_u$ and the artificial grouping factor to the model frame,
4. set up, but do not fit this model with a call to `lmer` with option `doFit=FALSE`, and finally
5. overwrite the design matrices for the random intercept of the artificial grouping factor with $\boldsymbol{Z}_p$.

Some complications arise if the `by-`, `allPen-` or `varying`-options are used, but these steps remain basically the same. The modified unfitted model is given to `lmer_finalize` or `glmer_finalize` for calling the optimization C-code.

4.3 Why use the TP-basis?

The following flaws of the TP-basis that is used as the default in amer are often mentioned:

- it has an undesirable one-to-one mapping between the smoothness/differentiability of the fit (TP of degree $p \Rightarrow$ fit is $p-1$-time continuous differentiable), and the nullspace of the penalty (TP of degree $p \Rightarrow$ nullspace is a p-degree polynomial). This is different from, e.g., penalized B-Spline fits, where the order of the difference penalty that determines the nullspace of the penalty can be specified independently from the order of the spline bases which determine the smoothness of the fit.
- the unbounded support (to the right of the knot) of the truncated polynomials means that the values of the basis function can potentially become huge.
- the columns in $\boldsymbol{Z}_p$ containing the truncated polynomials are severely collinear, especially for closely spaced knots

Why did I use it nevertheless? For one, similar collinearity in $\boldsymbol{Z}_p$ is present for all other spline bases I am aware of after the mixed model reparameterization (2) described in section 2.1. More importantly, for all other spline bases I am aware of, the reparameterized design $\boldsymbol{Z}_p$ contains no systematic zeroes at all even if the matrix of the original basis functions $\boldsymbol{B}$ is sparse, while $\boldsymbol{Z}_p$ for the TP-basis is about 50% zeroes. This means that amer can take advantage of the sparse matrix operations in lme4 when `tp` or `tpU` is used, but not for any other bases.

4.4 Writing your own basis-generating function

It is fairly easy to implement your own basis generating function for use in amer. Such a function only has to fullfill the following criteria:

- It has to have at least the arguments
  - `x`, a `numeric` variable used for the smooth function,
  - `by`, a `factor` variable (default: `NULL`),
  - `allPen`, (a `logical`),

  - `diag`, (a `logical`),
  - `varying`, a `numeric` variable (default: `NULL`).

- It has to return a list with
  - an entry named `X`, which contains the matrix X_u without the intercept column (this can be a matrix with zero columns)
  - an entry named `Z`, which contains the matrix Z_p
  - an attribute `call`, which contains the result of `expand.call`

You don't have to worry about the reparametrization described in 2.1 - the utility function `reparameterizeDesign` creates $\boldsymbol{X}_u$ and $\boldsymbol{Z}_p$ for a given design matrix $\boldsymbol{B}$ and the associated penalty matrix bmK. The technical details of splitting up $\boldsymbol{X}_u$ and $\boldsymbol{Z}_p$ for a possible `by` variable, naming the columns in $\boldsymbol{X}_u$ etc. are performed by the utility function `expandBasis`.

As an example, let's add a variant of the TP-Basis to `amer`'s repertoire – let's say we want to get rid of the undesirable one-to-one mapping between the smoothness/differentiability of the fit and the null-space of the penalty of the TP-basis. We implement a simple basis-generating function `tp2` that lets us specify the dimensionality of the nullspace so that, for a TP-spline basis of degree p without intercept (see above), we can specify the degree of the global polynomial that is unpenalized[2]. Let's call this option `dimU`. If we set `dimU`$= p$, this corresponds to the conventional TP-Penalty. If `dimU`$< p$, columns containing global polynomials that would be in $\boldsymbol{X}_u$ for the conventional TP-Penalty are put in $\boldsymbol{Z}_p$ instead.

The following code implements a rough draft of the idea, with the default for using a quadratic TP-Basis ($p = 2$) (s.t. the fitted function is continuously differentiable, i.e. has no kinks) while penalizing deviations from linearity (`dimU`$=$ 1):

```
> tp2 <- function(x, p = 2, k = 15, dimU = 1, by = NULL,
+                   allPen = FALSE, diag = FALSE, varying = NULL,
+                   knots = quantile(x,
+                           probs = (2:(k - p + 1))/(k - p + 3)))
+ {
+         #dim. of nullspace can't be larger than p of TP-basis:
+         stopifnot(dimU <= p)
+
+         #always need this for the call attribute of the returned value:
```

[2]This basis is available im `amer` as `tpU`.

```
+         call <- as.list(expand.call())
+         call$knots <- knots
+
+         #global polynomial trends:
+         X <- outer(x, 0:p, "^")
+
+         #truncated power basis functions
+         Z <- outer(x, knots, "-")^p * outer(x, knots, ">")
+
+         #matrix of basis functions
+         B <- cbind(X, Z)
+
+         #the penalty matrix K is the identity matrix,
+         #        with (dimU+1)-leading zeroes
+         # (dimU + 1 because of the intercept column...)
+         K <- diag(ncol(B))
+         K[cbind(1:(dimU + 1), 1:(dimU + 1))] <- 0
+
+         #let reparameterizeDesign do the dirty work
+         D <- reparameterizeDesign(B, K)
+
+         #return X_u and Z_p
+         res <- list(X = D$X, Z = D$Z)
+         attr(res, "call") <- as.call(call)
+         return(res)
+ }
```

We can now use this function to fit a continuously differentiable function with penalized deviations from linearity to the dog data, but we have to tell `amer` to look for smooth terms called `tp2` in the `basisGenerators`-option:

```
> d4 <- amer(y ~ -1 + group + tp2(time, k = 6, p = 2, dimU = 1,
+     by = group) + (1 | dog), data = dog, basisGenerators = c("tp2"))

> print(d4, corr = F)

Additive mixed model fit by REML
Formula: y ~ -1 + group + tp2(time, k = 6, p = 2, dimU = 1, by = group) +      (1 |
   Data: dog
 AIC BIC logLik deviance REMLdev
 373 422   -172      340     345
Random effects:
 Groups        Name        Variance Std.Dev.
 dog           (Intercept) 2.49e-01 4.99e-01
```

```
 f.time.group4 tp2         1.17e-01 3.41e-01
 f.time.group3 tp2         1.47e+00 1.21e+00
 f.time.group2 tp2         1.01e-11 3.18e-06
 f.time.group1 tp2         1.85e+01 4.30e+00
 Residual                  1.50e-01 3.88e-01
Number of obs: 252, groups: dog, 36; f.time.group4, 5; f.time.group3, 5; f.time.grou

Fixed effects:
                 Estimate Std. Error t value
group1              4.215      0.481    8.76
group2              3.553      0.164   21.61
group3              4.571      0.290   15.76
group4              4.034      0.202   19.98
time.group1.fx1    -0.286      1.162   -0.25
time.group2.fx1     0.110      0.155    0.71
time.group3.fx1    -2.284      0.658   -3.47
time.group4.fx1    -0.727      0.366   -1.99
```

Figure 7 shows a comparison of the fit with the `tp2`-function to the fit of a conventional linear TP-basis.

The following code generates figure 7:

```
> par(mfrow = c(1, 2))
> plotF(d1, legend = "topleft", auto.layout = F)
> plotF(d4, legend = "none", auto.layout = F)
```

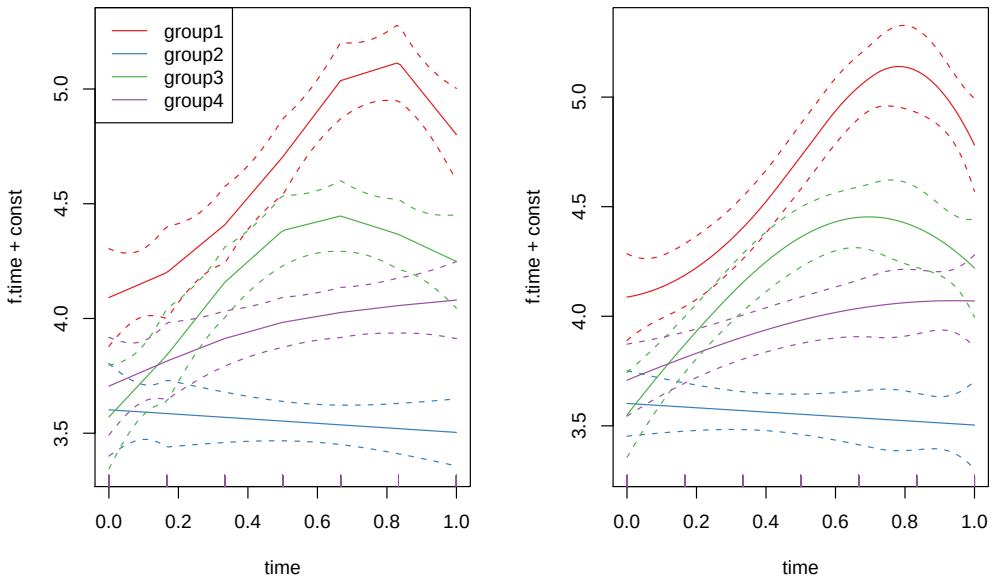

Figure 7: Comparison of the results for `tp(degree=1)` (left panel) and `tp2(degree=2, dimU=1)` (right panel) for the `dog` data.

5 Open Issues

A to-do list for developing amer further:

- approximate frequentist CI's for smooths with `allPen=TRUE` (should be easy, only modify `fctV`)
- 2D-smooths (will mean major re-working of most utility functions called by `amerSetup` as well as `getF/PlotF`)
- implementing (parametric/wild/...) bootstrap-CIs (could use `lme4:::refit` and Ben Bolker's `mer.sim`, maybe implement Kauermann/Claeskens/Opsomer (2008))
- CIs for functions based on profile likelihood methods from `lme4a`?
- implementing other spline bases, e.g. for cyclic functions

References

Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2009a. URL http://CRAN.R-project.org/package=Matrix. R package version 0.999375-30.

Douglas Bates and Martin Maechler. *lme4: Linear mixed-effects models using S4 classes*, 2009b. URL http://CRAN.R-project.org/package=lme4. R package version 0.999375-31.

E. Berndt. *The Practice of Econometrics: Classical and Contemporary*. Addison - Wesley, 1991.

C. Crainiceanu, D. Ruppert, and M.P. Wand. Bayesian analysis for penalized spline regression using WinBUGS. *Journal of Statistical Software*, 14(14):1–24, 2005.

P.H.C. Eilers and B.D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89–121, 1996.

J.E. Grizzle and D.M. Allen. Analysis of growth and dose response curves. *Biometrics*, 25:357–381, 1969.

T. Kneib. *Mixed model based inference in structured additive regression*. PhD thesis, LMU München, 2006.

D. Ruppert, M.P. Wand, and R.J. Carroll. *Semiparametric Regression*. Cambridge University Press, 2003.

S. Wood. *Generalized Additive Models*. Chapman and Hall, 2006a.

S. Wood. *mgcv: Multiple Smoothing Parameter Estimation by GCV or UBRE*, 2006b. URL http://www.maths.bath.ac.uk/~sw283/.