Universität Innsbruck

BAMLSS

Bayesian Additive Models for Location Scale and Shape (and Beyond)

Nikolaus Umlauf

http://eeecon.uibk.ac.at/~umlauf/

Overview

- Introduction
- Distributional regression
- Lego toolbox
- R package **bamlss**
- Example

Introduction

A **not** complete list of software packages dealing with Bayesian regression models:

- **bayesm**, univariate and multivariate, SUR, multinomial logit, ...
- **bayesSurv**, survival regression, ...
- **MCMCpack**, linear regression, logit, ordinal probit, probit, Poisson regression, ...
- **MCMCglmm**, generalized linear mixed models (GLMM).
- **spikeSlabGAM**, Bayesian variable selection, model choice, in generalized additive mixed models (GAMM), ...
- **gammSlice**, generalized additive mixed models (GAMM).
- **BayesX**, structured additive distributional regression (STAR), ...
- **INLA**, generalized additive mixed models (GAMM), ...
- **WinBUGS**, **JAGS**, **STAN**, general purpose sampling engines.

$\vdots$

Introduction

Most Bayesian software packages provide support for the estimation of so called mixed models (random effects), i.e., incorporating linear predictors of the form

$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{U}\boldsymbol{\gamma},$$

where $\mathbf{X}\boldsymbol{\beta}$ are fixed effects, e.g., $p(\boldsymbol{\beta}) \propto$ const, and $\mathbf{U}\boldsymbol{\gamma}$ are the random effects, $\boldsymbol{\gamma} \sim N(\mathbf{0}, \mathbf{Q}(\boldsymbol{\tau}^2))$.

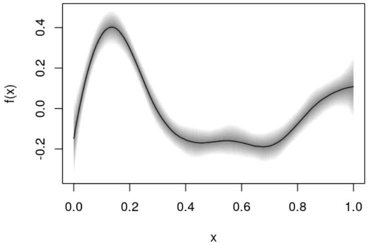
Few Bayesian software packages provide support for the estimation of semiparametric regression models with structured additive predictor

$$\eta = f_1(\mathbf{x}) + \ldots + f_J(\mathbf{x}),$$

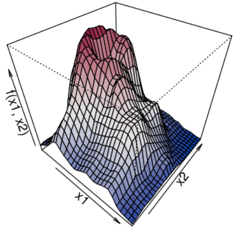
where f_j are possibly smooth functions and $\mathbf{x}$ represents a generic vector of all nonlinear modeled covariates.

Introduction

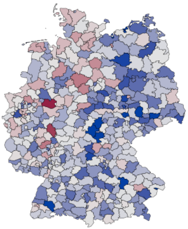
Nonlinear effects of continuous covariates



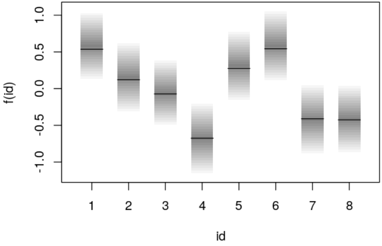
Two-dimensional surfaces



Spatially correlated effects f(x) = f(s)



Random intercepts f(x) = f(id)

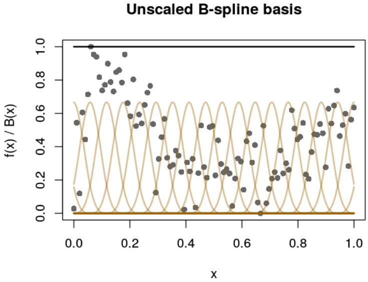


STAR Models

The vector of function evaluations $\mathbf{f}_j = (f_j(\mathbf{x}_1), \ldots, f_j(\mathbf{x}_n))$ of the $i = 1, \ldots, n$ observations is given by

$$\mathbf{f}_j = f_j(\mathbf{X}_j, \boldsymbol{\beta}_j) = \mathbf{X}\boldsymbol{\beta}_j,$$

with $\mathbf{X}_j$ as the design matrix and $\boldsymbol{\beta}_j$ are unknown regression coefficients. Form of $\mathbf{X}_j$ only depends on the functional type chosen, e.g., using B-splines:



Introduction

Basis function approach, penalized least squares:

$$\text{PLS}(\boldsymbol{\beta}, \boldsymbol{\lambda}) = ||\mathbf{y} - \boldsymbol{\eta}||^2 + \lambda_1 \boldsymbol{\beta}_1' \mathbf{K}_1 \boldsymbol{\beta}_1 + \ldots + \lambda_J \boldsymbol{\beta}_J' \mathbf{K}_J \boldsymbol{\beta}_J.$$

A general Prior for $\boldsymbol{\beta}$ in the corresponding Bayesian approach

$$p(\boldsymbol{\beta}_j) \propto \left(\frac{1}{\tau_j^2}\right)^{rk(\mathbf{K}_j)/2} \exp\left(-\frac{1}{2\tau_j^2}\boldsymbol{\beta}_j' \mathbf{K}_j \boldsymbol{\beta}_j\right),$$

τ_j^2 variance parameter, governs the smoothness of f_j.

Structure of $\mathbf{K}_j$ also depends on the type of covariates and on assumptions about smoothness of f_j.

The variance parameter τ_j^2 is equivalent to the inverse smoothing parameter in a frequentist approach.

Introduction

However, any basis function representation can be transformed into a mixed model representation

$$\mathbf{f}_j = \mathbf{X}_j \boldsymbol{\beta}_j = \mathbf{X}_j(\tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}} + \tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}}) = \dot{\mathbf{X}}\tilde{\boldsymbol{\beta}} + \dot{\mathbf{U}}\tilde{\boldsymbol{\gamma}},$$

with fixed effects $\tilde{\boldsymbol{\beta}}$ and random effects $\tilde{\boldsymbol{\gamma}} \sim N(\mathbf{0}, \tau^2\mathbf{I})$.

So the number of software packages that can estimate semiparametric models is actually quite large.

The number of different models that can be fit with these engines is even larger.

Introduction

The basic ideas are:

- Design a framework that makes it (a) easy to use different estimation engines and (b) fit models with a **structured additive predictor**.
- Therefore, we need to employ symbolic descriptions that do **not** restrict to any specific type of model and term structure.
- I.e., the aim is to use specialized/optimized engines to apply Bayesian **structured additive distributional regression** a.k.a. Bayesian additive models for location scale and shape (**BAMLSS**) and beyond.
- The approach should have **maximum flexibility/extendability**, also concerning functional types.

Distributional regression

Within this framework any parameter of a population distribution may be modeled by explanatory variables

$$y \sim \mathcal{D}\left(h_1(\theta_1) = \eta_1,\ h_2(\theta_2) = \eta_2, \ldots,\ h_K(\theta_K) = \eta_K\right),$$

where $\mathcal{D}$ denotes any parametric distribution available for the response variable.

Each parameter is linked to a structured additive predictor

$$h_k(\theta_k) = \eta_k = \eta_k(\mathbf{x}; \boldsymbol{\beta}_k) = f_{1k}(\mathbf{x}; \boldsymbol{\beta}_{1k}) + \ldots + f_{J_k k}(\mathbf{x}; \boldsymbol{\beta}_{J_k k}),$$

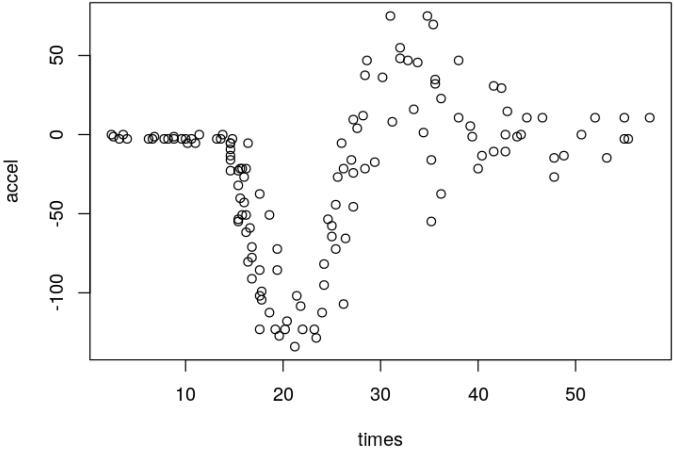
$j = 1, \ldots, J_k$ and $k = 1, \ldots, K$ and $h_k(\cdot)$ are known monotonic link functions.

The observations y_i are assumed to be independent and conditional on a pre-specified parametric density $f(y_i; \theta_{i1}, \ldots, \theta_{iK})$.

Distributional regression

Example: Head acceleration in a simulated motorcycle accident

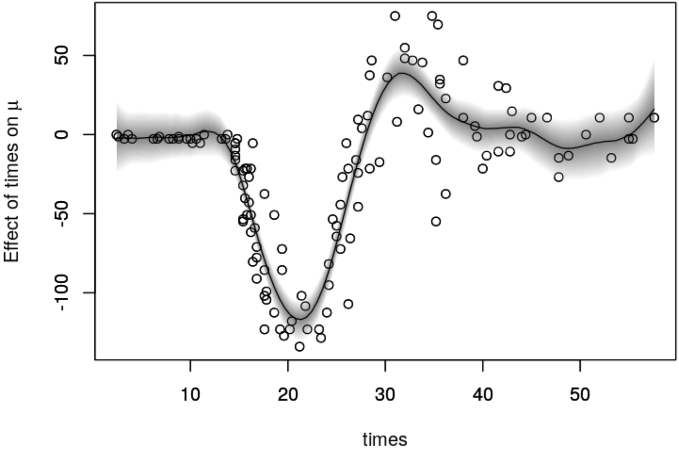
$$\texttt{accel} \sim N(\mu, \sigma^2).$$



Distributional regression

Example: Head acceleration in a simulated motorcycle accident

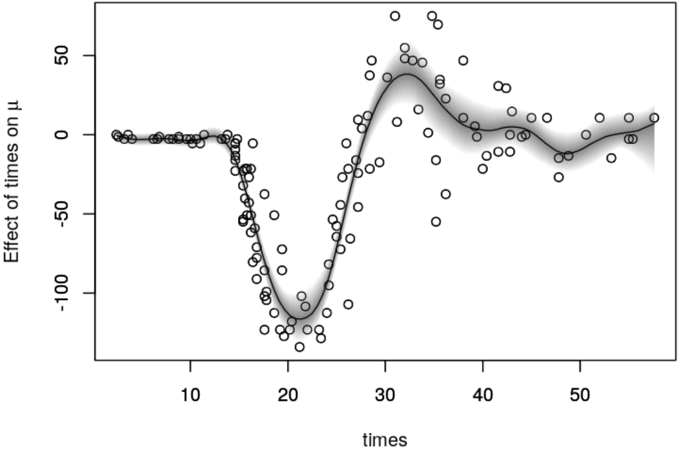
$$\texttt{accel} \sim N(\mu = f(\texttt{times}), \, log(\sigma^2) = \beta_0).$$



Distributional regression

Example: Head acceleration in a simulated motorcycle accident

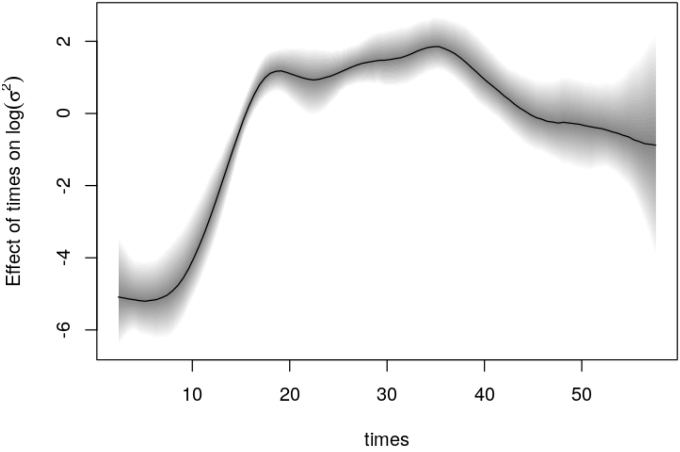
$$\texttt{accel} \sim N(\mu = f(\texttt{times}), \, log(\sigma^2) = f(\texttt{times})).$$



Distributional regression

Example: Head acceleration in a simulated motorcycle accident

$$\texttt{accel} \sim N(\mu = f(\texttt{times}),\, log(\sigma^2) = f(\texttt{times})).$$



A conceptional Lego toolbox
Families

Families specify the details of models.

Required details may differ from engine to engine, however, to fully "understand" a distribution we need the following:

- The density function.
- The distribution function.
- The quantile function.
- Link function(s).
- A random number generator.
- First and second derivatives of the log-likelihood (expectations).

So implementing a "new" distribution means creating a new family (object), including the minimum specifications required by the estimating engine(s).

A conceptional Lego toolbox
Priors

For simple linear effects $\mathbf{X}_{jk}\boldsymbol{\beta}_{jk}$, a common choice is $p(\boldsymbol{\beta}_{jk}) \propto \textit{const}$.

For the smooth terms, a general setup is obtained by

$$p(\boldsymbol{\beta}_{jk}) \propto \left(\frac{1}{\tau_{jk}^2}\right)^{rk(\mathbf{K}_{jk})/2} \exp\left(-\frac{1}{2\tau_{jk}^2}\boldsymbol{\beta}_{jk}^\top \mathbf{K}_{jk}\boldsymbol{\beta}_{jk}\right),$$

where $\mathbf{K}_{jk}$ is a quadratic penalty matrix that shrinks parameters towards zero or penalizes too abrupt jumps between neighboring parameters, e.g., for random effects $\mathbf{K}_{jk} = \mathbf{I}$.

Weakly informative inverse Gamma hyperprior

$$p(\tau_{jk}^2) = \frac{b_{jk}^{a_{jk}}}{\Gamma(a_{jk})}(\tau_{jk}^2)^{-(a_{jk}+1)} \exp(-b_{jk}/\tau_{jk}^2).$$

with $a_{jk} = b_{jk} = 0.001$,

A conceptional Lego toolbox
Model fitting

The main building block of regression model algorithms is the probability density function $f(\mathbf{y}|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$.

Estimation typically requires to evaluate

$$\begin{aligned} \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) &= \sum_{i=1}^{n} \log f(y_i; \theta_{i1} = h_1^{-1}(\eta_{i1}(\mathbf{x}_i, \boldsymbol{\beta}_1)), \ldots \\ &\qquad\qquad \ldots, \theta_{iK} = h_K^{-1}(\eta_{iK}(\mathbf{x}_i, \boldsymbol{\beta}_K))), \end{aligned}$$

with $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_K^\top)^\top$ and $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_K)$.

The log-posterior

$$\log p(\boldsymbol{\vartheta}; \mathbf{y}, \mathbf{X}) \propto \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) + \sum_{k=1}^{K} \sum_{j=1}^{J_k} \left\{ \log p_{jk}(\boldsymbol{\vartheta}_{jk}) \right\},$$

where, e.g., $\boldsymbol{\vartheta}_{jk} = (\boldsymbol{\beta}_{jk}^\top, (\boldsymbol{\tau}_{jk}^2)^\top)^\top$ (frequentist, penalized log-likelihood).

A conceptional Lego toolbox
Model fitting

Gradient based algorithms require the first derivative or score vector. Within the Bayesian formulation the resulting score vector is

$$\mathbf{s}(\boldsymbol{\beta}) = \frac{\partial \log\, p(\boldsymbol{\vartheta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\beta}} = \frac{\partial \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\beta}} + \sum_{k=1}^{K} \sum_{j=1}^{J_k} \left\{ \frac{\partial \log\, p_{jk}(\boldsymbol{\beta}_{jk})}{\partial \boldsymbol{\beta}} \right\},$$

The first order partial derivatives of the log-likelihood w.r.t. $\boldsymbol{\beta}$ can be further fragmented

$$\frac{\partial \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\beta}_k} = \frac{\partial \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\eta}_k} \frac{\partial \boldsymbol{\eta}_k}{\partial \boldsymbol{\beta}_k} = \frac{\partial \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \boldsymbol{\eta}_k} \frac{\partial \boldsymbol{\eta}_k}{\partial \boldsymbol{\beta}_k},$$

since $\theta_{ik} = h_k^{-1}(\eta_{ik}(\mathbf{x}_i, \boldsymbol{\beta}_k))$.

A conceptional Lego toolbox

Model fitting

Applying, e.g., Newton-Raphson requires the Hessian, entries $\mathbf{H}_{ks}(\boldsymbol{\beta})$

$$\frac{\partial^2 \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\beta}_k \partial \boldsymbol{\beta}_s^\top} = \left(\frac{\partial \boldsymbol{\eta}_s}{\partial \boldsymbol{\beta}_s}\right)^\top \frac{\partial^2 \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\eta}_k \partial \boldsymbol{\eta}_s^\top} \frac{\partial \boldsymbol{\eta}_k}{\partial \boldsymbol{\beta}_k} \underbrace{+ \frac{\partial \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\eta}_k} \frac{\partial^2 \boldsymbol{\eta}_k}{\partial^2 \boldsymbol{\beta}_k}}_{\text{if } k=s},$$

$k = 1, \dots, K$ and $s = 1, \dots, K$. Again, chain rule gives

$$\frac{\partial^2 \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\eta}_k \partial \boldsymbol{\eta}_s^\top} = \frac{\partial \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\theta}_k} \frac{\partial^2 \boldsymbol{\theta}_k}{\partial \boldsymbol{\eta}_k \partial \boldsymbol{\eta}_s^\top} + \frac{\partial^2 \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\theta}_k \partial \boldsymbol{\theta}_s^\top} \frac{\partial \boldsymbol{\theta}_k}{\partial \boldsymbol{\eta}_k} \frac{\partial \boldsymbol{\theta}_s}{\partial \boldsymbol{\eta}_s}.$$

Conventional updating scheme

$$\boldsymbol{\beta}^{(t+1)} = U(\boldsymbol{\beta}^{(t)}) = \boldsymbol{\beta}^{(t)} - \mathbf{H}\left(\boldsymbol{\beta}^{(t)}\right)^{-1} \mathbf{s}\left(\boldsymbol{\beta}^{(t)}\right),$$

feasable, but computationally still a bit unhandy.

A conceptional Lego toolbox

Model fitting

Fortunately, partitioned updating is possible

$$\begin{array}{rcl}
\boldsymbol{\beta}_1^{(t+1)} & = & U_1(\boldsymbol{\beta}_1^{(t)}, \boldsymbol{\beta}_2^{(t)}, \ldots, \boldsymbol{\beta}_K^{(t)}) \\
\boldsymbol{\beta}_2^{(t+1)} & = & U_2(\boldsymbol{\beta}_1^{(t+1)}, \boldsymbol{\beta}_2^{(t)}, \ldots, \boldsymbol{\beta}_K^{(t)}) \\
 & \vdots & \\
\boldsymbol{\beta}_K^{(t+1)} & = & U_K(\boldsymbol{\beta}_1^{(t+1)}, \boldsymbol{\beta}_2^{(t+1)}, \ldots, \boldsymbol{\beta}_K^{(t)}),
\end{array}$$

which yields

$$\boldsymbol{\beta}_k^{(t+1)} = U_k(\boldsymbol{\beta}_k^{(t)} | \cdot) = \boldsymbol{\beta}_k^{(t)} - \mathbf{H}_{kk}\left(\boldsymbol{\beta}_k^{(t)}\right)^{-1} \mathbf{s}\left(\boldsymbol{\beta}_k^{(t)}\right).$$

Can be further partitioned for each function within parameter block k.

A conceptional Lego toolbox
Model fitting

Using a basis function approach, derive PM-estimates with iteratively reweighted least squares (IWLS)

$$\boldsymbol{\beta}_{jk}^{(t+1)} = (\mathbf{X}_{jk}^\top \mathbf{W}_{kk} \mathbf{X}_{jk} + \mathbf{G}_{jk})^{-1} \mathbf{X}_{jk}^\top \mathbf{W}_{kk} (\mathbf{z}_k - \boldsymbol{\eta}_{k,-j}^{(t)}),$$

with $\mathbf{G}_{jk} = \tau_{jk}^{-2} \mathbf{K}_{jk}$ and working observations

$$\mathbf{z}_k = \boldsymbol{\eta}_k^{(t)} + \mathbf{W}_{kk}^{-1} \mathbf{u}_k^{(t)},$$

where $\mathbf{W}_{kk} = -\mathrm{diag}(\partial^2 \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) / \partial \boldsymbol{\eta}_k \partial \boldsymbol{\eta}_k^\top)$ and $\mathbf{u}_k = \partial \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) / \partial \boldsymbol{\eta}_k$.

Depending on the type of algorithm different weights are used, e.g., $\mathbf{W}_{kk} = E\left(-\partial^2 \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) / \partial \boldsymbol{\eta}_k \partial \boldsymbol{\eta}_k^\top\right)$.

A conceptional Lego toolbox

Model fitting

MCMC simulation

- Random walk Metropolis, symmetric $q(\boldsymbol{\beta}_{jk}^{\star}|\boldsymbol{\beta}_{jk}^{(t)})$.
- Derivative based MCMC, second order Taylor series expansion centered at the last state $p(\boldsymbol{\beta}_{jk}^{\star}|\cdot)$ yields $N(\boldsymbol{\mu}_{jk}^{(t)}, \boldsymbol{\Sigma}_{jk}^{(t)})$ proposal with precision matrix

$$\left(\boldsymbol{\Sigma}_{jk}^{(t)}\right)^{-1} = -\mathbf{H}_{kk}\left(\boldsymbol{\beta}_{jk}^{(t)}\right)$$

and mean

$$\boldsymbol{\mu}_{jk}^{(t)} = \boldsymbol{\beta}_{jk}^{(t)} - \mathbf{H}_{kk}\left(\boldsymbol{\beta}_{jk}^{(t)}\right)^{-1}\mathbf{s}\left(\boldsymbol{\beta}_{jk}^{(t)}\right).$$

Metropolis-Hastings acceptance probability

$$\alpha\left(\boldsymbol{\beta}_{jk}^{\star}|\boldsymbol{\beta}_{jk}^{(t)}\right) = \min\left\{\frac{p(\boldsymbol{\beta}_{jk}^{\star}|\cdot)q(\boldsymbol{\beta}_{jk}^{(t)}|\boldsymbol{\beta}_{jk}^{\star})}{p(\boldsymbol{\beta}_{jk}^{(t)}|\cdot)q(\boldsymbol{\beta}_{jk}^{\star}|\boldsymbol{\beta}_{jk}^{(t)})}, 1\right\}.$$

A conceptional Lego toolbox
Model fitting

- Again, using a basis function approach, simplified Metropolis-Hastings based on IWLS proposals:

$$\boldsymbol{\mu}_{jk}^{(t)} = \boldsymbol{\Sigma}_{jk}^{(t)} \mathbf{X}_{jk}^{\top} \mathbf{W}_{kk} \left\{ \mathbf{z}_k - \boldsymbol{\eta}_{k,-j}^{(t)} \right\},$$

  and precision matrix

$$\left(\boldsymbol{\Sigma}_{jk}^{(t)} \right)^{-1} = \mathbf{X}_{jk}^{\top} \mathbf{W}_{kk} \mathbf{X}_{jk} + \mathbf{G}_{jk},$$

  resulting multivariate normal proposal

$$\boldsymbol{\beta}_{jk}^{\star} \sim N(\boldsymbol{\mu}_{jk}^{(t)}, \boldsymbol{\Sigma}_{jk}^{(t)}).$$

- Other sampling schemes, e.g., slice sampling, NUTS, t-walk, …?!

A conceptional Lego toolbox

Summary

The following “lego bricks” are repeatedly used within BAMLSS candidate algorithms:

- The density function $f(y; \theta_i = h_1^{-1}(\eta_1(\mathbf{x}, \boldsymbol{\beta}_1)), \ldots, \theta_K = h_K^{-1}(\eta_K(\mathbf{x}, \boldsymbol{\beta}_K)))$,
- link functions $h_k(\cdot)$,
- the first order derivatives $\frac{\partial \log\, p(\boldsymbol{\vartheta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\beta}_k}$ and $\frac{\partial \boldsymbol{\eta}_k}{\partial \boldsymbol{\beta}_k}$,
- the second order derivatives $\frac{\partial^2 \log\, p(\boldsymbol{\vartheta}; \mathbf{y}, \mathbf{X})}{\partial \boldsymbol{\beta}_k \partial \boldsymbol{\beta}_s^\top}$,
- derivatives for log-priors $\frac{\partial \log\, p_{jk}(\boldsymbol{\vartheta}_{jk})}{\partial \boldsymbol{\vartheta}_{jk}}$.

A conceptional Lego toolbox

Algorithm

A simple generic algorithm for BAMLSS models:

while(eps > ε & t < maxit) {
    for(k in 1:K) {
        for(j in 1:J[k]) {
            Compute $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta}_{\mathtt{k}} - \mathbf{f}_{\mathtt{jk}}$.
            Obtain new $(\boldsymbol{\beta}^{\star}_{\mathtt{jk}}, (\tau^2_{\mathtt{jk}})^{\star})^{\top} = \mathtt{U}_{\mathtt{jk}}(\mathbf{X}_{\mathtt{jk}}, \mathbf{y}, \tilde{\boldsymbol{\eta}}, \boldsymbol{\beta}^{[\mathtt{t}]}_{\mathtt{jk}}, (\tau^2_{\mathtt{jk}})^{[\mathtt{t}]})$.
            Update $\boldsymbol{\eta}_{\mathtt{k}}$.
        }
    }
    t = t + 1
    Compute new eps.
}

Functions $\mathtt{U}_{\mathtt{jk}}(\cdot)$ could either return proposals from a MCMC sampler or updates from an optimizing algorithm.

R package bamlss

The package is available at

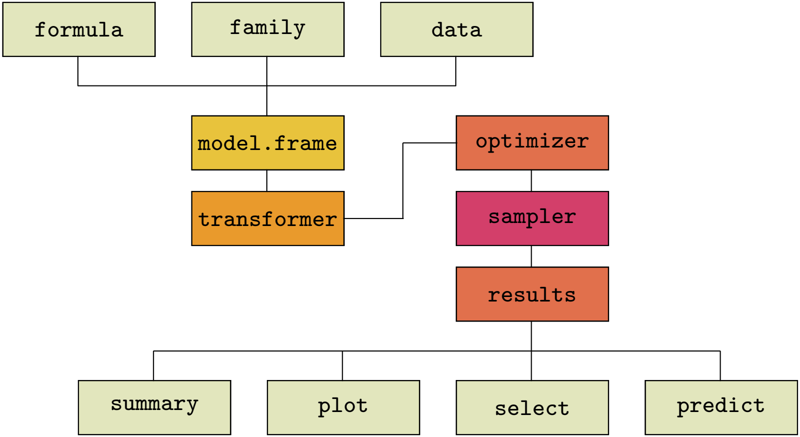
`https://R-Forge.R-project.org/projects/BayesR/`

In R, simply type

```
R> install.packages("bamlss",
+   repos = "http://R-Forge.R-project.org")
```

R package bamlss

Building blocks



In principle, the setup does not restrict to any specific type of engine (Bayesian or frequentist).

Based on Wilkinson and Rogers (1973) a typical model description in R has the form

$$\texttt{response} \sim \texttt{x1 + x2}.$$

Using structured additive predictors we need generic descriptors for smooth/random terms, creating the type of term/basis we want to incorporate (model frame). The recommended R package **mgcv** (Wood 2006) has a pretty set up, e.g.

$$\texttt{response} \sim \texttt{x1 + x2 + s(z1) + s(z2, z3)}$$

$$\texttt{response} \sim \texttt{x1 + x2 + s(z1, bs = "ps")}.$$

R package bamlss

Symbolic descriptions

In the context of distributional regression we need formula extensions for multiple parameters. One convenient way to specify, e.g., the parameters of a normal model is:

```
list(
  response ~ x1 + x2 + s(z1) + s(z2),
  sigma ~ x1 + x2 + s(z1)
)
```

A four parameter example:

```
list(
  response ~ x1 + x2 + s(z1) + s(z2),
  sigma2 ~ x1 + x2 + s(z1),
  nu ~ s(z1),
  tau ~ s(z2)
)
```

R package bamlss

Symbolic descriptions

Hierarchical structures:

```
list(
  response ~ x1 + x2 + s(z1) + s(id1),
  id1 ~ x3 + s(z3) + s(id2),
  id2 ~ s(z4),
  sigma2 ~ x1 + x2 + s(z1),
  nu ~ s(z1) + s(id1),
  tau ~ s(z2)
)
```

Categorical responses:

```
list(
  response ~ x1 + x2 + s(z1) + s(z2),
  ~ x1 + x2 + s(z1) + s(z3)
)
```

R package bamlss

The model frame

Parsing the necessary model frame is based on **mgcv** infrastructures. In addition, the parser allows to define special user defined terms.

```
bamlss.frame(formula, data = NULL, family = "gaussian",
  weights = NULL, subset = NULL, offset = NULL,
  na.action = na.omit, contrasts = NULL, ...)
```

Creates the model frame, i.e., all necessary matrices to set up a model.

```
R> f <- list(accel ~ s(times), sigma ~ s(times))
R> bf <- bamlss.frame(f, data = mcycle, family = "gaussian")
```

R package bamlss

```
R> print(bf)

'bamlss.frame' structure:
  ..$ call
  ..$ model.frame
  ..$ formula
  ..$ family
  ..$ terms
  ..$ x
  .. ..$ mu
  .. .. ..$ formula
  .. .. ..$ fake.formula
  .. .. ..$ terms
  .. .. ..$ model.matrix
  .. .. ..$ smooth.construct
  .. ..$ sigma
  .. .. ..$ formula
  .. .. ..$ fake.formula
  .. .. ..$ terms
  .. .. ..$ model.matrix
  .. .. ..$ smooth.construct
  ..$ y
  .. ..$ accel
```

R package bamlss

Workflow example

JAGS

```
R> bf$samples <- with(bf, JAGS(x, y, family))
R> summary.bamlss(bf)
R> plot.bamlss(bf)
```

BayesX

```
R> f <- list(
+   accel ~ sx(times),
+   sigma ~ sx(times)
+ )
R> bf <- bamlss.frame(f, data = mcycle, family = "gaussian")
R> bf$samples <- with(bf, BayesX(x, y, family))
R> summary.bamlss(bf)
R> plot.bamlss(bf)
```

(Note: currently not working.)

R package bamlss

Available building blocks

Type	Function
Parser	`bamlss.frame()`
Transformer	`bamlss.engine.setup()`, `randomize()`
Optimizer	`bfit()`, `opt()`, `cox.mode()`
Sampler	`GMCMC()`, `JAGS()`, `STAN()`, `BayesX()`, `cox.mcmc()`
Results	`results.bamlss.default()`

If new engines are implemented, one only needs to exchange the building block functions.

R package bamlss
Available families

Work in progress . . . (+ note that not all families are available for all implemented engines yet)

BCCG	cens	cloglog	cox
beta	dagum	lognormal	quant
betazi	dirichlet	multinomial	t
betazi	gamma	mvn	truncgaussian
betazoi	gaussian	mvt	truncgaussian2
binomial	gaussian2	negbin	weibull
bivlogit	gengamma	pareto	zinb
bivprobit	invgaussian	poisson	zip

Families with ending 2 represent alternative parametrizations.

R package bamlss
Family constructor

Basic setup of **bamlss** families, e.g., for $N(\mu, \sigma^2)$:

```
list(
    family, names, links,
    d(y, par),
    p(y, par),
    q(y, par),
    r(y, par),
    score = list(
       mu(y, par),
       sigma(y, par)
    ),
    hess = list(
       mu(y, par),
       sigma(y, par)
    )
)
```

Extendable, e.g., specify the engines to be used, too.

R package bamlss
Wrapper function

To ease the workflow, a wrapper function for the available engines is provided:

```
bamlss(formula, family = "gaussian",
  data = NULL, start = NULL, transform = NULL,
  optimizer = NULL, sampler = NULL, results = NULL,
  cores = NULL, combine = TRUE, ...)
```

Standard extractor and plotting functions are provided:

`summary()`, `plot()`, `fitted()`, `residuals()`, `predict()`, `coef()`, `logLik()`, `DIC()`, `samples()`, ...

Example

Cox-regression for fire emergengy response times

The *London Fire Brigade* is one of the largest in the world. Collects huge amounts of data, e.g., incidents records from dwelling fire:

`http://data.london.gov.uk/dataset/london-fire-brigade-incident-records`

Reponse times of emergency calls, how can these be improved?

Example taken from:

Taylor BM, Rowlingson B (2015). ***spatsurv**: An R Package for Bayesian Inference with Spatial Survival Models.* (to appear)
URL: `http://www.lancaster.ac.uk/staff/taylorb1/preprints/spatsurv.pdf`

Example

Data set of the first two quarters of 2015 emergency calls from dwelling fire, available in **bamlss**:

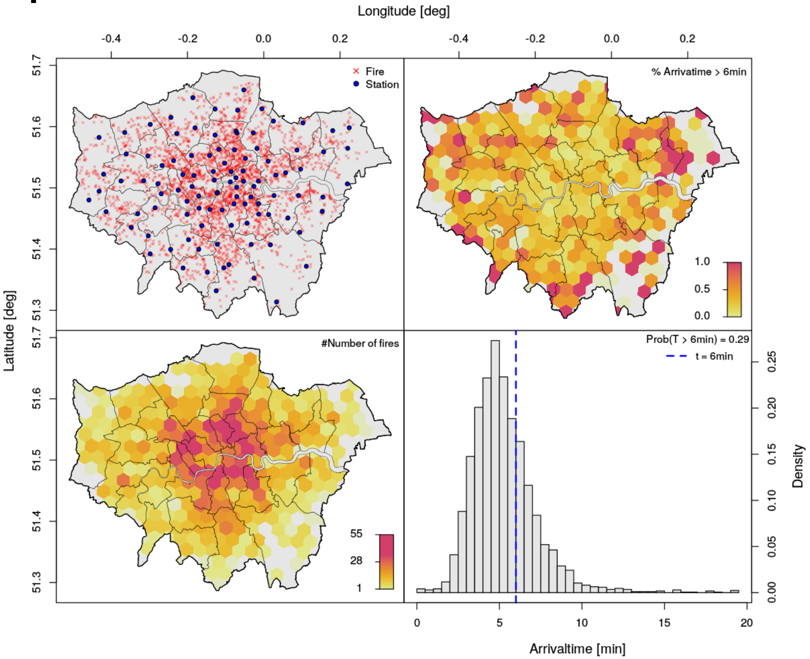
```
R> data("LondonFire", package = "bamlss")
R> nrow(LondonFire)
```

```
[1] 5838
```

Consists of the `"SpatialPointsDataFrame"` named `LondonFire` and the actual 2015 fire station locations `LondonFStations`, as well as the `"SpatialPolygons"` `LondonBoroughs` and `LondonBoundaries`.

```
R> plot(LondonFire, col = "red")
R> plot(LondonFStations, col = "blue", add = TRUE)
R> plot(LondonBoroughs, add = TRUE)
```

Example



Longitude [deg]
-0.4
-0.2
0.0
0.2
Latitude [deg]
51.3
51.4
51.5
51.6
51.7
Fire
Station
% Arrivatime > 6min
1.0
0.5
0.0
#Number of fires
55
28
1
Prob(T > 6min) = 0.29
t = 6min
Density
0.00
0.05
0.10
0.15
0.20
0.25
0
5
10
15
20
Arrivaltime [min]

Example

Cox-model

We are interested in the drivers of the time it takes until the first fire engine arrives after the emergengy call.

The hazard of an event (fire engine arrives) at time t can be described with a relative additive risk model of the form:

$$\lambda(t) = \exp\left(\eta(t)\right) = \exp\left(\eta_\lambda(t) + \eta_\gamma\right),$$

i.e., a model for the instantaneous arrival rate conditional on the engine having not arrived before time t.

The probability that the engine will arrive on the scene after time t is

$$S(t) = Prob(T > t) = \exp\left(-\int_0^t \lambda(u)du\right).$$

Example

For NR and MCMC we need the log-likelihood of the continuous time Cox-model

$$\ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{n} \left(\delta_i \eta_{i,\gamma} - \int_0^{t_i} \exp(\eta_{i,\lambda}(u) du) \right)$$

Assuming a basis function approach, the score vector for the time-dependent part is

$$\mathbf{s}\left(\boldsymbol{\beta}_\lambda\right) = \boldsymbol{\delta}^\top \mathbf{X}_\lambda(\mathbf{t}) - \sum_{i=1}^{n} \exp(\eta_{i,\gamma}) \left(\int_0^{t_i} \exp(\eta_{i,\lambda}(u)) \mathbf{x}_i(u) du \right).$$

The elements of the Hessian w.r.t. $\boldsymbol{\beta}_\lambda$ are

$$\mathbf{H}\left(\boldsymbol{\beta}_\lambda\right) = -\sum_{i=1}^{n} \exp\left(\eta_{i,\gamma}\right) \int_0^{t_i} \exp(\eta_{i,\lambda}(u)) \boldsymbol{x}_{i,\lambda}(u) \boldsymbol{x}_{i,\lambda}^\top(u) du.$$

Example

The integrals need to be computed numerically, e.g., using the trapezoidal rule we "only" need to set up a time grid, lets say with 100 equidistant points within $[0, t_i]$

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}_1^\top \\ \vdots \\ \mathbf{g}_n^\top \end{pmatrix}, \quad \text{with} \quad \mathbf{g}_i = (0, \ldots, t_i)^\top,$$

to construct the evaluated $\lambda(t)$ matrix with

$$\hat{\eta}_\lambda(\mathbf{G}) = \begin{pmatrix} \sum_{j=1}^{J_\lambda} f_j(x_{1j}(g_{10})) & \ldots & \sum_{j=1}^{J_\lambda} f_j(x_{1j}(g_{1t_i})) \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^{J_\lambda} f_j(x_{nj}(g_{n0})) & \ldots & \sum_{j=1}^{J_\lambda} f_j(x_{nj}(g_{nt_i})) \end{pmatrix}.$$

Example

Fortunately, the time-constant part is a bit easier. Results in IWLS backfitting/proposal scheme with

$$\mathbf{z} = \boldsymbol{\eta}_{\gamma} + \mathbf{W}^{-1}\mathbf{u}$$

with diagonal matrix

$$\mathbf{W} = \mathit{diag}(\exp(\boldsymbol{\eta}_{\gamma}) \cdot \mathbf{I})$$

and

$$\mathbf{u} = \boldsymbol{\delta} - \exp(\boldsymbol{\eta}_{\gamma}) \cdot \mathbf{I}.$$

Here, diagonal matrix $\mathbf{I}$ represents the integrals for all individuals.

Optimizer and sampler implemented in function `cox.mode()` and `cox.mcmc()`.

Example

For the emergency call model, we use the following additive predictors

$$\eta_\lambda = f_1(\texttt{arrivaltime}) + f_2(\texttt{arrivaltime}, \texttt{lon}, \texttt{lat})$$

and

$$\begin{aligned}\eta_\gamma &= \beta_0 + f_1(\texttt{fsintens}) + f_2(\texttt{daytime}) + \\ &\quad f_3(\texttt{lon}, \texttt{lat}) + f_4(\texttt{daytime}, \texttt{lon}, \texttt{lat}).\end{aligned}$$

In R we set up the model by

```
R> f <- list(
+   Surv(arrivaltime) ~ ti(arrivaltime) + ti(arrivaltime,lon,lat),
+   gamma ~ s(fsintens) + ti(daytime,bs="cc") + ti(lon,lat) +
+       ti(daytime,lon,lat,bs=c("cc","cr"),d=c(1,2))
+ )

R> firemodel <- bamlss(f, data = LondonFire, family = "cox",
+   subdivisions = 100, n.iter = 12000, burnin = 2000,
+   thin = 10, cores = 8, maxit = 1000)
```

Example

```
R> summary(firemodel)

Call:
bamlss(formula = f, family = "cox", data = LondonFire, cores = 7,
    subdivisions = 100, nu = 0.01, n.iter = 4000, burnin = 2000,
    thin = 10, maxit = 3000)
---
Family: cox
Link function: lambda = log, gamma = identity
*---
Formula lambda:
---
Surv(arrivaltime) ~ ti(arrivaltime) + ti(arrivaltime, lon, lat)
-
Smooth terms:
                                   Mean      2.5%       50%     97.5%
ti(arrivaltime).tau21         2.904e-01 7.555e-02 2.414e-01 7.795e-01
ti(arrivaltime).edf           1.097e+01 8.684e+00 1.088e+01 1.360e+01
ti(arrivaltime).alpha         3.722e-01 4.032e-05 2.124e-01 1.000e+00
ti(arrivaltime,lon,lat).tau21 3.143e-07 2.810e-07 3.134e-07 3.500e-07
ti(arrivaltime,lon,lat).edf   3.500e+01 3.500e+01 3.500e+01 3.500e+01
ti(arrivaltime,lon,lat).alpha 1.134e-01 5.330e-33 4.015e-03 1.000e+00
```

Example

```
                                parameters
ti(arrivaltime).tau21                0.132
ti(arrivaltime).edf                  9.926
ti(arrivaltime).alpha                   NA
ti(arrivaltime,lon,lat).tau21        5.350
ti(arrivaltime,lon,lat).edf         57.097
ti(arrivaltime,lon,lat).alpha           NA
---
Formula gamma:
---
gamma ~ s(fsintens) + ti(daytime, bs = "cc") + ti(lon, lat) +
  ti(daytime, lon, lat, bs = c("cc", "cr"), d = c(1, 2))
-
Parametric coefficients:
               Mean    2.5%     50%   97.5% parameters
(Intercept) -0.9425 -0.9768 -0.9427 -0.9088     -0.926
-
```

Example

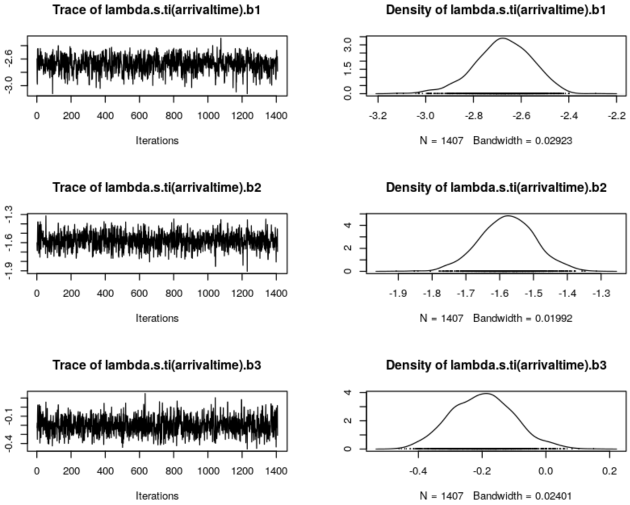
```
Smooth terms:

... not shown ...

---
Sampler summary:
-
DIC = 13365.95 logLik = -6628.066 logPost = 3307.896
pd = 109.8167
---
Optimizer summary:
-
edf = 170.6112 logLik = -6533.596 logPost = -8017.541
```

Example

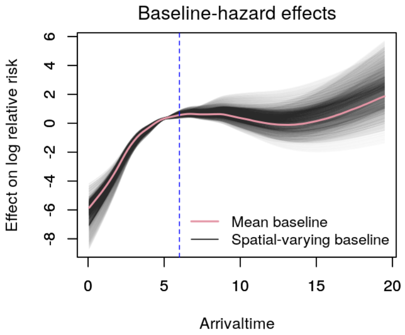
```
R> plot(firemodel, which = "samples")
```

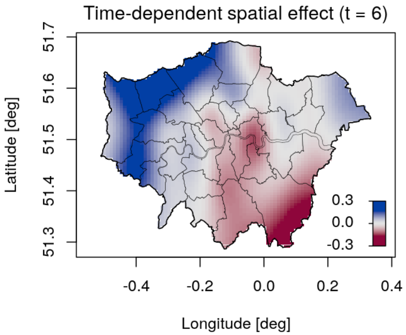


Example

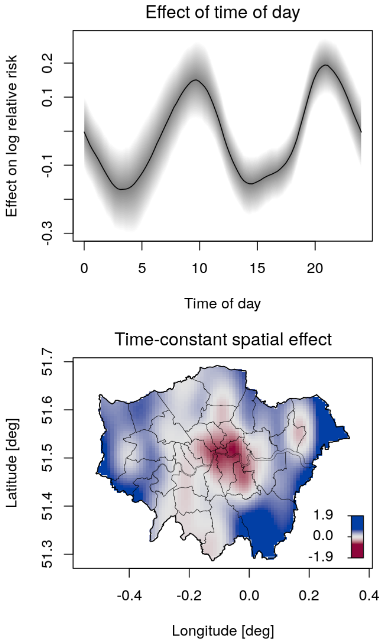
```
R> plot(firemodel, model = "lambda", term = "s(arrivaltime)")

R> predict(firemodel, newdata = nd,
+   model = "lambda", term = "s(lon,lat,arrivaltime)")
```





Example



Effect of time of day
Effect on log relative risk
0.2
0.1
-0.1
-0.3
0
5
10
15
20
Time of day
Effect of fire station intensity
Effect on log relative risk
1.0
0.0
-1.0
-2.0
0
500
1000
1500
2000
Fire station intensity
Time-constant spatial effect
Latitude [deg]
51.7
51.6
51.5
51.4
51.3
1.9
0.0
-1.9
-0.4
-0.2
0.0
0.2
0.4
Longitude [deg]
Prob(T > 6)
Latitude [deg]
51.7
51.6
51.5
51.4
51.3
0.4
0.2
0.0
-0.4
-0.2
0.0
0.2
0.4
Longitude [deg]

Thank you!!!

Klein N, Kneib T, Klasen S, Lang S (2014). *Bayesian Structured Additive Distributional Regression for Multivariate Responses.* Journal of the Royal Statistical Society: Series C (Applied Statistics), pp. URL `http://dx.doi.org/10.1111/rssc.12090`

Lang S, Umlauf N, Wechselberger P, Harttgen K, Kneib T (2013): Multilevel structured additive regression. *Statistics and Computing*, 24(2), 223–238.

Umlauf N, Adler D, Kneib T, Lang S, Zeileis A (2015). *Structured additive regression models: An R interface to **BayesX***. Journal of Statistical Software, 63(21):1–46, 2015. URL `http://CRAN.R-project.org/package=R2BayesX`

Rigby RA, Stasinopoulos DM (2005). *Generalized Additive Models for Location, Scale and Shape (with Discussion)*. Applied Statistics 54, 507–554.

Wood SN (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC, Boca Raton.

Wood SN (2011). ***mgcv**: GAMs with GCV/AIC/REML Smoothness Estimation and GAMMs by PQL*. R package version 1.7-6. URL `http://CRAN.R-project.org/package=mgcv`