A Conceptional Lego Toolbox for Bayesian Distributional Regression Models

Nikolaus Umlauf, Nadja Klein, Meike Köhler, Sonja Greven, Achim Zeileis

http://eeecon.uibk.ac.at/~umlauf/
Example: Head acceleration in a simulated motorcycle accident

\[ \text{accel} \sim N(\mu, \sigma^2). \]
Introduction

Example: Head acceleration in a simulated motorcycle accident

\[ \text{accel} \sim N(\mu = f(\text{times}), \log(\sigma^2) = \beta_0). \]
Introduction

Example: Head acceleration in a simulated motorcycle accident

$$\text{accel} \sim N(\mu = f(\text{times}), \log(\sigma^2) = f(\text{times}))$$.
Example: Head acceleration in a simulated motorcycle accident

\[ \text{accel} \sim \mathcal{N}(\mu = f(\text{times}), \log(\sigma^2) = f(\text{times})). \]
Distributional regression
Model structure

Any parameter of a population distribution $\mathcal{D}$ may be modeled by explanatory variables

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

Each parameter is linked to a structured additive predictor

$$h_k(\theta_k) = \eta_k = \eta_k(x; \beta_k) = f_{1k}(x; \beta_{1k}) + \ldots + f_{J_kk}(x; \beta_{J_kk}),$$

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

Functional types

Nonlinear effects of continuous covariates

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

Two-dimensional surfaces

Random intercepts $f(x) = f(id)$
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.

...
Basic ideas

- Design framework to fit models with **different estimation engines** (Bayesian or frequentist).
- **Integration** of new and existing code, as well as interfacing, as easy as possible.
- Symbolic descriptions that do **not** restrict to any specific type of model and term structure.
- Specialized/optimized engines to apply Bayesian **structured additive distributional regression** a.k.a. Bayesian additive models for location scale and shape (**BAMLSS**) and beyond.
- **Maximum flexibility/extendability**, also concerning functional types.
A conceptional Lego toolbox

Common priors

For simple linear effects $X_{jk} \beta_{jk}$: $p(\beta_{jk}) \propto \text{const}$.

For the smooth terms:

$$p(\beta_{jk}) \propto \left( \frac{1}{\tau_{jk}^2} \right)^{rk(K_{jk})/2} \exp \left( -\frac{1}{2\tau_{jk}^2} \beta_{jk}^\top K_{jk} \beta_{jk} \right),$$

where $K_{jk}$ is a quadratic penalty matrix.

Priors $p(\tau_{jk}^2)$: IG, half-Cauchy, half-normal, uniform priors, etc.
A conceptional Lego toolbox
Model fitting

The main building block of regression model algorithms is the probability density function $f(y|\theta_1, \ldots, \theta_K)$. Estimation typically requires to evaluate

$$\ell(\beta; y, X) = \sum_{i=1}^{n} \log f(y_i; \theta_{i1} = h_1^{-1}(\eta_{i1}(x_i, \beta_1)), \ldots, \theta_{iK} = h_K^{-1}(\eta_{iK}(x_i, \beta_K))),$$

with $\beta = (\beta_1^\top, \ldots, \beta_K^\top)^\top$ and $X = (X_1, \ldots, X_K)$. The log-posterior

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

where, e.g., $\vartheta_{jk} = (\beta_{jk}^\top, (\tau_{jk}^2)^\top)^\top$ (frequentist, penalized log-likelihood).
A conceptional Lego toolbox

Model fitting

Posterior mode estimation, fortunately, partitioned updating is possible

\[ \beta_{1}^{(t+1)} = U_1(\beta_{1}^{(t)}, \beta_{2}^{(t)}, \ldots, \beta_{K}^{(t)}) \]
\[ \beta_{2}^{(t+1)} = U_2(\beta_{1}^{(t+1)}, \beta_{2}^{(t)}, \ldots, \beta_{K}^{(t)}) \]
\[ \vdots \]
\[ \beta_{K}^{(t+1)} = U_K(\beta_{1}^{(t+1)}, \beta_{2}^{(t+1)}, \ldots, \beta_{K}^{(t)}) \]

E.g., Newton-Raphson type updating

\[ \beta_{k}^{(t+1)} = U_k(\beta_{k}^{(t)} | \cdot) = \beta_{k}^{(t)} - H_{kk} \left( \beta_{k}^{(t)} \right)^{-1} s \left( \beta_{k}^{(t)} \right). \]

Can be further partitioned for each function within parameter block \( k \). Moreover, using a basis function approach yields IWLS updates

\[ \beta_{jk}^{(t+1)} = (X_{jk}^\top W_{kk} X_{jk} + \tau_{jk}^{-2} K_{jk})^{-1} X_{jk}^\top W_{kk} (z_k - \eta_{k,-j}^{(t)}). \]
A conceptional Lego toolbox

Model fitting

MCMC simulation

- Random walk Metropolis, symmetric $q(\beta_{jk}^* | \beta_{jk}^{(t)})$.
- Derivative based MCMC, second order Taylor series expansion centered at the last state $p(\beta_{jk}^* | \cdot)$ yields $\mathcal{N}(\mu_{jk}^{(t)}, \Sigma_{jk}^{(t)})$ proposal with

$$
\left(\Sigma_{jk}^{(t)}\right)^{-1} = -H_{kk} \left(\beta_{jk}^{(t)}\right)
$$

$$
\mu_{jk}^{(t)} = \beta_{jk}^{(t)} - H_{kk} \left(\beta_{jk}^{(t)}\right)^{-1} s \left(\beta_{jk}^{(t)}\right).
$$

Metropolis-Hastings acceptance probability

$$
\alpha \left(\beta_{jk}^* | \beta_{jk}^{(t)}\right) = \min \left\{ \frac{p(\beta_{jk}^* | \cdot)q(\beta_{jk}^{(t)} | \beta_{jk}^*)}{p(\beta_{jk}^{(t)} | \cdot)q(\beta_{jk}^* | \beta_{jk}^{(t)})}, 1 \right\}.
$$

- Other sampling schemes, e.g., slice sampling, NUTS, t-walk, . . . ?!
A conceptional Lego toolbox

Model fitting

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

**For log-posterior (likelihood):**
- Density function of response distribution $\mathcal{D}$,
- link functions $h_k(\cdot)$,
- priors.

**For e.g., IWLS fitting or derivative based MCMC:**
- First and second order derivatives of log-posterior,
- often can be put together from derivatives of log-density, link functions and log-priors.
A conceptional Lego toolbox

Algorithm

A simple generic algorithm for BAMLSS models:

```python
while(eps > ε & t < maxit) {
    for(k in 1:K) {
        for(j in 1:J[k]) {
            Compute \( \tilde{\eta} = \eta_k - f_{jk} \).
            Obtain new \((\beta_{jk}^*, (\tau_{jk}^2)^*)\) \(= U_{jk}(X_{jk}, y, \tilde{\eta}, \beta_{jk}^{[t]}, (\tau_{jk}^2)^{[t]})\).
            Update \( \eta_k \).
        }
    }
    t = t + 1
    Compute new eps.
}
```

Functions \( U_{jk}(\cdot) \) 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")
In principle, the setup does not restrict to any specific type of engine (Bayesian or frequentist).
R package bamlss
Available building blocks

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser</td>
<td><code>bamlss.frame()</code></td>
</tr>
<tr>
<td>Transformer</td>
<td><code>bamlss.engine.setup()</code>, <code>randomize()</code></td>
</tr>
<tr>
<td>Optimizer</td>
<td><code>bfit()</code>, <code>opt()</code>, <code>cox.mode()</code>, <code>jm.mode()</code></td>
</tr>
<tr>
<td>Sampler</td>
<td><code>GMCMC()</code>, <code>JAGS()</code>, <code>STAN()</code>, <code>BayesX()</code>,</td>
</tr>
<tr>
<td></td>
<td><code>cox.mcmc()</code>, <code>jm.mcmc()</code></td>
</tr>
<tr>
<td>Results</td>
<td><code>results.bamlss.default()</code></td>
</tr>
</tbody>
</table>

To implement new engines, only the building block functions have to be exchanged.
R package bamlss

Available families

Work in progress ...

<table>
<thead>
<tr>
<th>Function</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta.bamlss()</td>
<td>Beta distribution</td>
</tr>
<tr>
<td>binomial.bamlss()</td>
<td>Binomial distribution</td>
</tr>
<tr>
<td>cnorm.bamlss()</td>
<td>Censored normal distribution</td>
</tr>
<tr>
<td>cox.bamlss()</td>
<td>Continuous time Cox-model</td>
</tr>
<tr>
<td>gaussian.bamlss()</td>
<td>Gaussian distribution</td>
</tr>
<tr>
<td>gamma.bamlss()</td>
<td>Gamma distribution</td>
</tr>
<tr>
<td>jm.bamlss()</td>
<td>Continuous time joint-model</td>
</tr>
<tr>
<td>multinomial.bamlss()</td>
<td>Multinomial distribution</td>
</tr>
<tr>
<td>mvn.bamlss()</td>
<td>Multivariate normal distribution</td>
</tr>
<tr>
<td>poisson.bamlss()</td>
<td>Poisson distribution</td>
</tr>
</tbody>
</table>

New families only require density, distribution, random number generator, quantile, score and hess functions.
R package bamlss

Wrapper function:

bamlss(list(accel ~ s(times), sigma ~ s(times)),
      family = "gaussian", data = mcycle, optimizer = bfit,
      sampler = GMCMC)

Standard extractor and plotting functions:

summary(), plot(), fitted(), residuals(), predict(), coef(),
logLik(), DIC(), samples(),...
Joint modeling of longitudinal and survival data

The hazard of an event at time $t$ can be described with a relative additive risk model of the form:

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

i.e., a model for the instantaneous event rate conditional on the event did not happen before time $t$.

The probability that an event will occur after time $t$ is

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

In a joint-model setting, additional longitudinal data available, e.g., a biomarker measured at regular/irregular intervals.
Example

Is the risk of an event associated with the longitudinal process?

\[ \lambda(t) = \exp(\eta(t)) = \exp(\eta_\lambda(t) + \eta_\gamma + \eta_\alpha(t) \cdot \eta_\mu(t)). \]

Assuming conditional independence, the corresponding log-likelihood is

\[ \ell(\beta; y, X) = \ell(\beta_{\text{surv}}; y_{\text{surv}}, X_{\text{surv}}) + \ell(\beta_{\text{long}}; y_{\text{long}}, X_{\text{long}}). \]

For estimation, compute:

- \( s(\beta_\lambda), s(\beta_\gamma), s(\beta_\mu), s(\beta_\sigma) \) and \( s(\beta_\alpha) \).
- \( H(\beta_\lambda), H(\beta_\gamma), H(\beta_\mu), H(\beta_\sigma) \) and \( H(\beta_\alpha) \).

I.e., posterior mode estimation with NR and derivative based MCMC.

Note, the log-likelihood and derivatives include integrals, which need to be computed numerically.
Example

In R, posterior mode estimates with function `jm.mode()`, MCMC sampling with function `jm.mcmc()`.

Both functions use the infrastructures provided by `bamlss.frame()`.

Formula for the JM

```r
R> f <- list(
+   Surv2(survtime, event, obs = y) ~ s(survtime),
+   gamma ~ s(x1),
+   mu ~ s(x2) + ti(obstime) + ti(id,bs="re") +
+       ti(id,obstime,bs=c("re","cr"),k=c(nlevels(d$data$id), 5)),
+   sigma ~ 1,
+   alpha ~ s(survtime)
+ )
```

The model is estimated with

```r
R> b <- bamlss(f, data = d, family = "jm",
+   timevar = "obstime", idvar = "id",
+   n.iter = 12000, burnin = 2000, thin = 40, cores = 4)
```
Example

```
R> plot(b, model = c("lambda", "alpha"))
```
Example

```
R> nd <- subset(d, id %in% c(1:20))
R> nd$fit <- predict(b, newdata = nd, model = "mu",
  +   term = c("s(x2)", "ti(obstime)", "ti(id)", "ti(id,obstime)"))
```
Example

R> plot(b, model = "alpha", which = "samples")
Thank you!!!


