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, . . .
- **MCMCgllm**, 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.

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

$$\eta = X\beta + U\gamma,$$

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

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

$$\eta = f_1(z) + \ldots + f_p(z) + x^T\beta,$$

where $f_j$ are possibly smooth functions and $z$ represents a generic vector of all nonlinear modeled covariates.
Introduction

Nonlinear effects of continuous covariates

Two-dimensional surfaces

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

Random intercepts
STAR Models

Within the basis function approach, the vector of function evaluations \( f_j = (f_j(z_1), \ldots, f_j(z_n)) \) of the \( i = 1, \ldots, n \) observations can be written in matrix notation

\[
f_j = Z_j \gamma_j,
\]

with \( Z_j \) as the design matrix, where \( \gamma_j \) are unknown regression coefficients. Form of \( Z_j \) only depends on the functional type chosen.
Penalized least squares:

\[ \text{PLS}(\gamma, \lambda) = \|y - \eta\|^2 + \lambda_1 \gamma_1' K_1 \gamma_1 + \ldots + \lambda_p \gamma_p' K_p \gamma_p. \]

A general Prior for \( \gamma \) in the corresponding Bayesian approach

\[ p(\gamma_j | \tau_j^2) \propto \exp \left( -\frac{1}{2\tau_j^2} \gamma_j' K_j \gamma_j \right), \]

\( \tau_j^2 \) variance parameter, governs the smoothness of \( f_j \).

Structure of \( K_j \) also depends on the type of covariates and on assumptions about smoothness of \( f_j \).

The variance parameter \( \tau_j^2 \) is equivalent to the inverse smoothing parameter in a frequentist approach.
However, any basis function representation can be transformed into a mixed model representation

\[ f_j = \mathbf{Z}_j \gamma_j = \mathbf{Z}_j (\tilde{\mathbf{X}} \beta + \tilde{\mathbf{U}} \tilde{\gamma}) = \mathbf{X} \beta + \mathbf{U} \tilde{\gamma}, \]

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

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

\[ y \sim D \left(g_1(\theta_1) = \eta_1, \ g_2(\theta_2) = \eta_2, \ldots, \ g_K(\theta_K) = \eta_K\right), \]

where \( D \) denotes any parametric distribution available for the response variable.

Each parameter is linked to a structured additive predictor

\[ g_k(\theta_k) = \eta_k = Z_{1k}\gamma_{1k} + \ldots + Z_{pk}\gamma_{pk} + X_k\beta_k, \quad k = 1, \ldots, K, \]

where \( g_k(\cdot) \) 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

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

Example: Head acceleration in a simulated motorcycle accident

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

Example: Head acceleration in a simulated motorcycle accident

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

Example: Head acceleration in a simulated motorcycle accident

\[ \text{accel} \sim \mathcal{N}(\mu = f(\text{times}), \log(\sigma^2) = f(\text{times})). \]
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 the linear part $\mathbf{X}\beta$, a common choice is $p(\beta) \propto \text{const.}$

For the smooth terms, a general setup is obtained by

$$p(\gamma_j | \tau_j^2) \propto \exp \left( -\frac{1}{2\tau_j^2} \gamma_j^\top \mathbf{K}_j \gamma_j \right),$$

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

Weakly informative inverse Gamma hyperprior

$$p(\tau_j^2) = \frac{b_j^{a_j}}{\Gamma(a_j)} (\tau_j^2)^{-(a_j+1)} \exp(-b_j/\tau_j^2).$$

with $a_j = b_j = 0.001$. 
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

$$\ell(\vartheta|y) = \sum_{i=1}^{n} \ln f(y_i|\theta_{i1} = h_1^{-1}(\eta_{i1}), \ldots, \theta_{iK} = h_K^{-1}(\eta_{iK})),$$

with $\vartheta = (\beta_1, \ldots, \beta_K, \gamma_1, \ldots, \gamma_K)^\top$.

The log-posterior

$$\ln p(\vartheta|y) = \ell(\vartheta|y) + \sum_{k=1}^{K} \sum_{j=1}^{p_k} \{ \ln p(\beta_{jk}|\tau_{jk}^2) + \ln p(\tau_{jk}^2) \},$$

where $\vartheta = (\beta_1, \ldots, \beta_K, \gamma_1, \ldots, \gamma_K, \tau_1^2, \ldots, \tau_K^2)^\top$ (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

\[ s(\vartheta) = \frac{\partial \ln p(\vartheta | y)}{\partial \vartheta} = \frac{\partial \ell(\vartheta | y)}{\partial \vartheta} + \sum_{k=1}^{K} \sum_{j=1}^{P_k} \left\{ \frac{\partial \ln p(\beta_{jk} | \tau_{jk}^2)}{\partial \vartheta} + \frac{\partial \ln p(\tau_{jk}^2)}{\partial \vartheta} \right\} , \]

The first order partial derivatives of the log-likelihood for \( \vartheta_k = (\beta_k, \gamma_k, \tau_k^2)^\top \), can be further fragmented

\[ \frac{\partial \ell(\vartheta | y)}{\partial \vartheta_k} = \frac{\partial \ell(\vartheta | y_i)}{\partial \eta_k} \frac{\partial \eta_k}{\partial \vartheta_k} = \frac{\partial \ell(\vartheta | y_i)}{\partial \theta_k} \frac{\partial \theta_k}{\partial \eta_k} \frac{\partial \eta_k}{\partial \vartheta_k} , \]

since \( \theta_{ik} = h_k^{-1}(\eta_{ik}(\vartheta_k)) \).
A conceptional Lego toolbox

Model fitting

Applying, e.g., a Newton-Raphson algorithm additionally requires the second derivatives

\[
\frac{\partial^2 \ell(\varphi | \mathbf{y})}{\partial \varphi_k \partial \varphi_s^\top} = \left( \frac{\partial \eta_s}{\partial \varphi_s} \right)^\top \frac{\partial^2 \ell(\varphi | \mathbf{y})}{\partial \eta_k \partial \eta_s^\top} \frac{\partial \eta_k}{\partial \varphi_k} + \frac{\partial \ell(\varphi | \mathbf{y})}{\partial \eta_k} \frac{\partial^2 \eta_k}{\partial \eta_k \partial \eta_k^\top}
\]

\[
\text{if } k = s
\]

PM-estimates with iteratively reweighted least squares (IWLS)

\[
\mathbf{z}_{[t]}^{[t]} = \eta_{[t]}^{[t]} + \left( \mathbf{W}_{[kk]}^{[t]} \right)^{-1} \mathbf{s}_{[t]}^{[t]},
\]

with \( \mathbf{s}_k = \frac{\partial \ell(\varphi | \mathbf{y})}{\partial \eta_k} \) and weights \( \mathbf{W}_{kk} = -\frac{\partial^2 \ell(\varphi | \mathbf{y})}{\partial \eta_k \partial \eta_k^\top} \).

Depending on the type of algorithm different weights are used, e.g., \( \mathbf{W}_{kk} = E \left( -\frac{\partial^2 \ell(\varphi | \mathbf{y})}{\partial \eta_k \partial \eta_k^\top} \right) \).
A conceptional Lego toolbox
Model fitting

MCMC simulation
- Metropolis-Hastings based on IWLS proposals:

\[ \mu_j = P_j^{-1}Z_j'W(z - \eta_{-j}) \quad P_j = Z_j'WZ_j + \frac{1}{\tau_j^2}K_j, \]

with working weights

\[ W = \text{diag} \left( E \left( -\frac{\partial^2 \ell}{\partial \eta_i^2} \right) \right), \]

and

\[ \gamma_j^{[t]} \sim N(\mu_j, P_j^{-1}). \]

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

Summary

The following quantities are repeatedly used within candidate algorithms:

- The density function \( f(y|\theta_1, \ldots, \theta_K) \).
- The first order derivatives \( \partial l(\vartheta|y)/\partial \theta_k, \partial \theta_k/\partial \eta_k \) and \( \partial \eta_k/\partial \vartheta_k \).
- Second order derivatives \( \partial^2 l(\vartheta|y)/\partial \eta_k \partial \eta_k^\top \).
- Derivatives for priors, e.g., \( \ln p(\gamma_{jk}|\tau_{jk}^2) \) and \( \ln p(\tau_{jk}^2) \).
A conceptional Lego toolbox

Algorithm

A simple generic algorithm for BAMLSS models:

```plaintext
while(\text{eps} > \varepsilon \& i < \text{maxit}) {
    for(k in 1:K) {
        for(j in 1:p) {
            Compute \( \eta_{-j}^{[k]} = \eta^{[k]} - f_j^{[k]} \).
            Obtain new \((\gamma_j^{[k]}, \tau_{-j}^{[k]})^\top = u_j^{[k]}(y, \eta_{-j}^{[k]}, z_j^{[k]}, \gamma_j^{[k]}, \tau_{-j}^{[k]}, \text{family}, k)\).
            Update \( \eta^{[k]} \).
        }
    }
    Compute new \text{eps}
}
```

Functions \( u_j^{[k]}(\cdot) \) 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
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>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser</td>
<td>parse.input.bayesr()</td>
</tr>
<tr>
<td>Transformer</td>
<td>randomize(), transformBUGS(), transformBayesX(), tranformBayesG()</td>
</tr>
<tr>
<td>Setup</td>
<td>setupJAGS(), jags2stan()</td>
</tr>
<tr>
<td>Engine</td>
<td>samplerBayesX(), samplerJAGS(), samplerSTAN(), samplerBayesG(), engine_stacker()</td>
</tr>
<tr>
<td>Results</td>
<td>resultsBayesX(), resultsBUGS(), resultsBayesG()</td>
</tr>
</tbody>
</table>

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)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BCCG</td>
<td>cens</td>
<td>cloglog</td>
<td>lognormal</td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>dagum</td>
<td>lognormal2</td>
<td>quant</td>
<td></td>
</tr>
<tr>
<td>betazi</td>
<td>dirichlet</td>
<td>multinomial</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>betazi</td>
<td>gamma</td>
<td>mvn</td>
<td>truncgaussian</td>
<td></td>
</tr>
<tr>
<td>betazoi</td>
<td>gaussian</td>
<td>mvt</td>
<td>truncgaussian2</td>
<td></td>
</tr>
<tr>
<td>binomial</td>
<td>gaussian2</td>
<td>negbin</td>
<td>weibull</td>
<td></td>
</tr>
<tr>
<td>bivlogit</td>
<td>gengamma</td>
<td>pareto</td>
<td>zinb</td>
<td></td>
</tr>
<tr>
<td>bivprobit</td>
<td>invgaussian</td>
<td>poisson</td>
<td>zip</td>
<td></td>
</tr>
</tbody>
</table>

Families with ending 2 represent alternative parametrizations.
To ease the workflow, a wrapper function for the available engines is provided:

\[
bamlss(formula, family = \text{gaussian}, data = \text{NULL},
    knots = \text{NULL}, weights = \text{NULL}, subset = \text{NULL},
    offset = \text{NULL}, na.action = \text{na.fail}, contrasts = \text{NULL},
    engine = \text{c("BayesG", "BayesX", "JAGS", "STAN")},
    cores = \text{NULL}, combine = \text{TRUE},
    n.iter = 12000, thin = 10, burnin = 2000,
    seed = \text{NULL}, \ldots)
\]

The function calls \text{xreg()} and returns an object of “bamlss” for which standard extractor and plotting functions are provided:

\[
\text{summary()}, \text{plot()}, \text{fitted()}, \text{residuals()}, \text{predict()}, \text{coef()}, \text{DIC()}, \text{samples()}, \ldots
\]
Example

Dynamical Statistical Forecast of Alpine Snow Amounts
Reto Stauffer, Jakob W. Messner, Achim Zeileis and Georg J. Mayr

Affected:
- Public transport.
- Winter tourism.
- Outdoor sportsmen.
- Residents & infrastructure.

Forecasts needed for:
- Risk assessments.
- Public warning.
- Road/railroad maintenance.
  - $+12h$ to few days in advance.

Challenges of rain/snow forecasting in complex terrain:
- Depends on various scales (global circulation $\rightarrow$ micro physics).
- Strongly modulated by local orography.
- Even high resolution NWP models do not resolve all important processes.
- Minor station density at high altitudes.
Overview of all precipitation observation stations in Tyrol.
Left panel: Spatial distribution.
Right panel: Station and topographic distribution.
Example

Basic concept:
Use anomalies to eliminate station dependence

\[
\text{obs} - \text{obs}_{\text{clim}} = \beta_0 + \beta_1 \cdot (\text{ens} - \text{ens}_{\text{clim}}) + \varepsilon.
\]

Corrected forecast:

\[
\hat{y} = \text{obs}_{\text{clim}} + \beta_0 + \beta_1 \cdot (\text{ens} - \text{ens}_{\text{clim}}).
\]

- **obs**: Observations.
- **obs\(_{\text{clim}}\)**: Climatology of observations.
- **ens**: Ensemble forecasts from an NWP model.
- **ens\(_{\text{clim}}\)**: Climatology of past ensemble forecasts.
- **\(\hat{y}\)**: Estimated, spatially corrected forecasts.
- **\(\varepsilon\)**: Statistical (unexplained) error.
Example

Daily precipitation observations 1970 – 2011:

Density

Fitted censored distribution

0.0 0.2 0.4 0.6 0.8 1.0

0 2 4 6 8 10 12 14

Daily √observations
Censored regression model: Latent Gaussian variable $y^*$ and observed response $y$ (square root of daily precipitation observations)

$$y^* \sim N(\mu, \sigma^2),$$

$$\mu = \eta_\mu, \quad \log(\sigma) = \eta_\sigma,$$

$$y = \max(0, y^*).$$

Predictors:

$$\eta = \beta_0 + f(y_{\text{day}}) + f(\text{alt}) + f(\text{lon}, \text{lat}).$$

Likelihood:

$$L(\vartheta | y) = \prod_{i=1}^n f(y_i | \vartheta, \sigma, z_i)^{I(y_i > 0)} \cdot P(y_i = 0 | z_i)^{I(y_i = 0)}.$$
Example

```
R> library("bamlss")
R> load("data/raindata.rda")
R> f <- list(
+   sqrt(obs) ~ s(yday,bs="cc") + s(alt) + s(lon,lat,k=50),
+   sigma ~ s(yday,bs="cc") + s(alt) + s(lon,lat,k=50)
+ )
R> rainmodel <- bamlss(f, data = dat,
+   family = gF(cens, left = 0),
+   method = c("backfitting", "MCMC"),
+   update = "iwls", propose = "iwls",
+   n.iter = 12000, burnin = 2000, thin = 10)
R> summary(rainmodel)
```
Example

Call:
bamlss(formula = f, family = gF(cens, left = 0), data = dat, ...)

Family: cens
Link function: mu = identity, sigma = log

---

Results for mu:
---

Formula:
sqrt(obs) ~ s(yday, bs = "cc") + s(alt) + s(lon, lat, k = 50)

Parametric coefficients:

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Sd</th>
<th>2.5%</th>
<th>50%</th>
<th>97.5%</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-0.166456</td>
<td>0.003707</td>
<td>-0.173706</td>
<td>-0.166600</td>
<td>-0.159903</td>
<td>1</td>
</tr>
</tbody>
</table>

Smooth effects variances:

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Sd</th>
<th>2.5%</th>
<th>50%</th>
<th>97.5%</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(yday)</td>
<td>4492.16</td>
<td>1794.10</td>
<td>2208.69</td>
<td>4108.18</td>
<td>8846.81</td>
<td>0.999</td>
</tr>
<tr>
<td>s(alt)</td>
<td>476.29</td>
<td>183.31</td>
<td>235.11</td>
<td>440.02</td>
<td>965.10</td>
<td>0.999</td>
</tr>
<tr>
<td>s(lon, lat)</td>
<td>273.88</td>
<td>41.31</td>
<td>204.93</td>
<td>270.01</td>
<td>367.54</td>
<td>0.997</td>
</tr>
</tbody>
</table>
Results for sigma:
---

Formula:
\sim s(yday, bs = "cc") + s(alt) + s(lon, lat, k = 50)

Parametric coefficients:

<table>
<thead>
<tr>
<th>Mean</th>
<th>Sd</th>
<th>2.5%</th>
<th>50%</th>
<th>97.5%</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.973318</td>
<td>0.001221</td>
<td>0.970857</td>
<td>0.973322</td>
<td>0.975730</td>
</tr>
</tbody>
</table>

Smooth effects variances:

<table>
<thead>
<tr>
<th>Mean</th>
<th>Sd</th>
<th>2.5%</th>
<th>50%</th>
<th>97.5%</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(yday)</td>
<td>506.40</td>
<td>231.64</td>
<td>234.84</td>
<td>460.71</td>
<td>1105.84</td>
</tr>
<tr>
<td>s(alt)</td>
<td>37.26</td>
<td>17.13</td>
<td>16.03</td>
<td>33.18</td>
<td>79.16</td>
</tr>
<tr>
<td>s(lon,lat)</td>
<td>110.74</td>
<td>17.77</td>
<td>82.03</td>
<td>109.02</td>
<td>154.17</td>
</tr>
</tbody>
</table>

---

DIC = 2.457e+06  N = 845321
Example

R> plot(rainmodel, term = c("s(yday)", "s(alt)"))
Example

R> plot(rainmodel, model = "mu", term = "s(lon,lat)"")
Example

R> plot(rainmodel, model = "sigma", term = "s(lon, lat)")

Effect on log(σ)
Example

R> p <- predict(rainmodel, model = "mu", newdata = nd, FUN = foo)
Example

Predictions for January 24:

| Location    | $P(y > 0|z)$ |
|-------------|-------------|
| Innsbruck   | 30.46%      |
| St.Anton    | 37.40%      |
| Galtür      | 37.61%      |
| Lienz       | 24.86%      |
| Sölden      | 30.38%      |
| Mayrhofen   | 32.28%      |
| Kitzbühel   | 38.29%      |


