universität
innsbruck

bamlss: A Lego toolbox for flexible regression models

Nikolaus Umlauf

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

Overview

Joint work with Achim Zeileis, Nadja Klein, Meike Köhler, Thorsten Simon and Stanislaus Stadlmann.

1. Introduction
2. Model specification
3. Model fitting
4. R package *bamlss*
5. Application

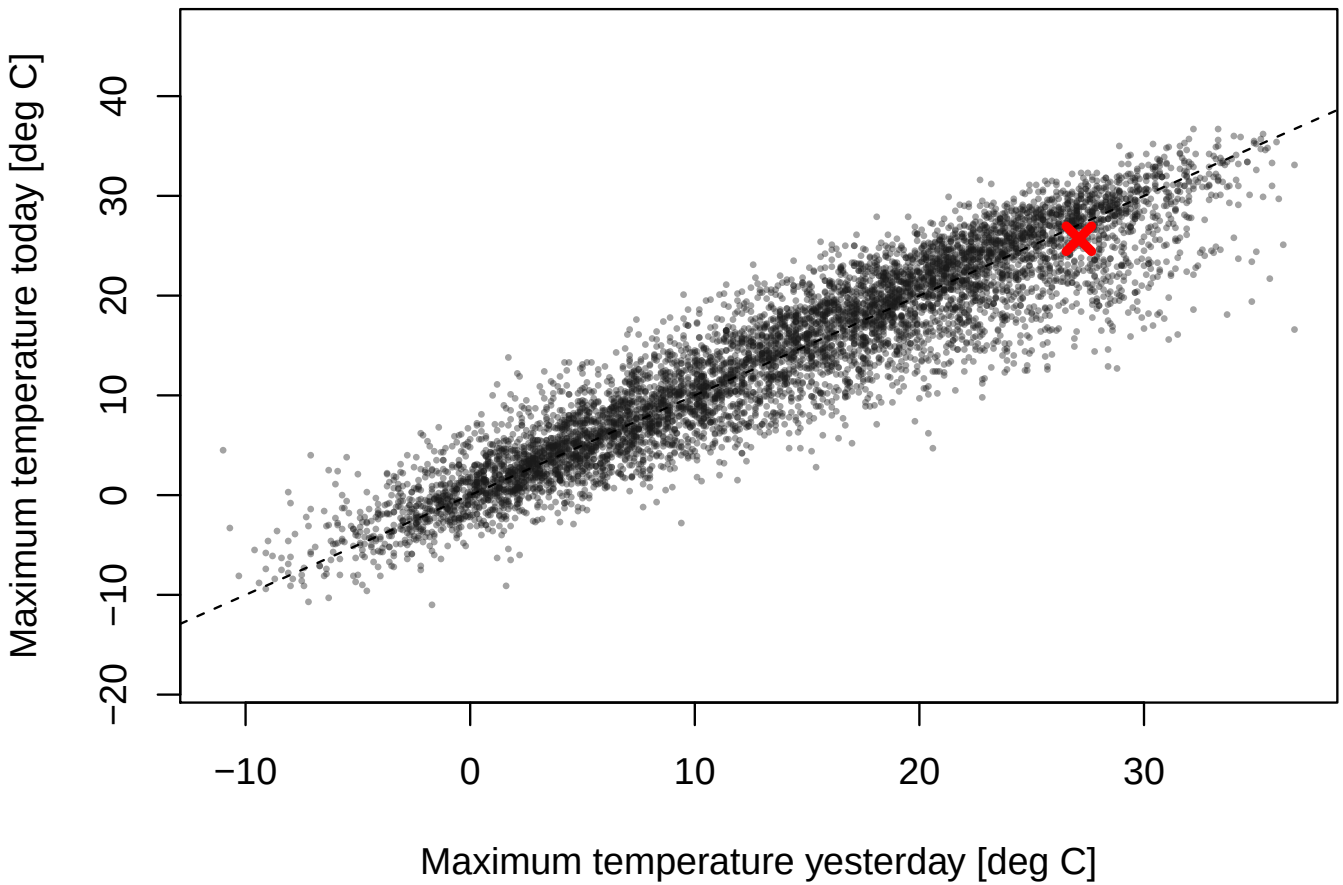
Introduction

- **Computational power** has tremendously increased.
- **Complicated inferential problems**, e.g., with MCMC simulation, possible on virtually any modern computer.
- To embed many **different approaches** suggested in literature and software, a **unified modeling architecture** for flexible regression models is particularly helpful.
- With the *bamlss* framework, implementing (new) algorithms, integration of already existing software, is relatively straightforward.
- The original idea came from flexible **Bayesian distributional regression** models.

Introduction

Linz daily maximum temperature data (2000/01-2018/09)

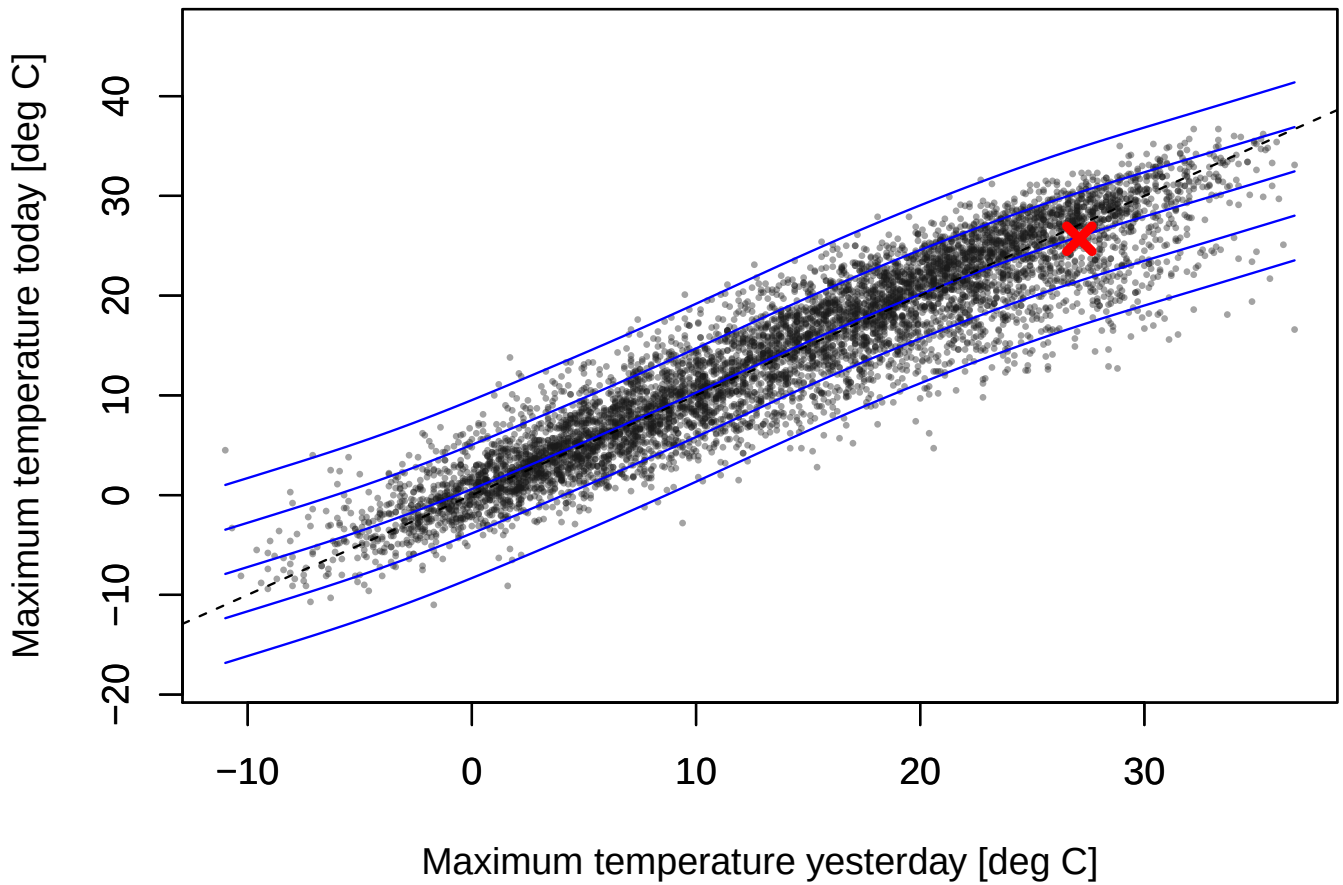
$$\mathtt{T} \sim N(\mu, \sigma^2).$$



Introduction

Linz daily maximum temperature data (2000/01-2018/09)

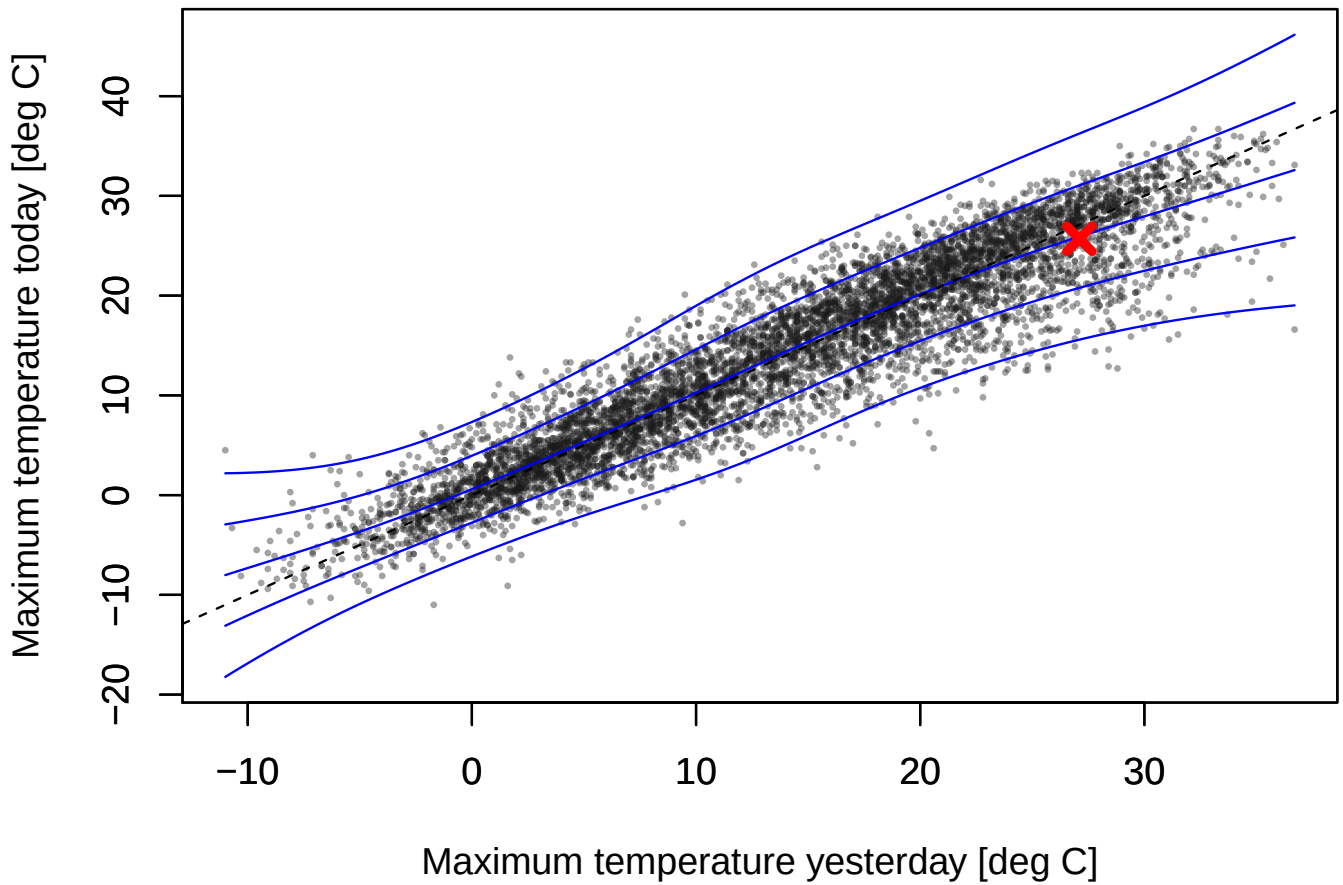
$$\mathrm{T} \sim N(\mu = f(\mathrm{T}_{t-1}),\ log(\sigma^2) = \beta_0).$$



Introduction

Linz daily maximum temperature data (2000/01-2018/09)

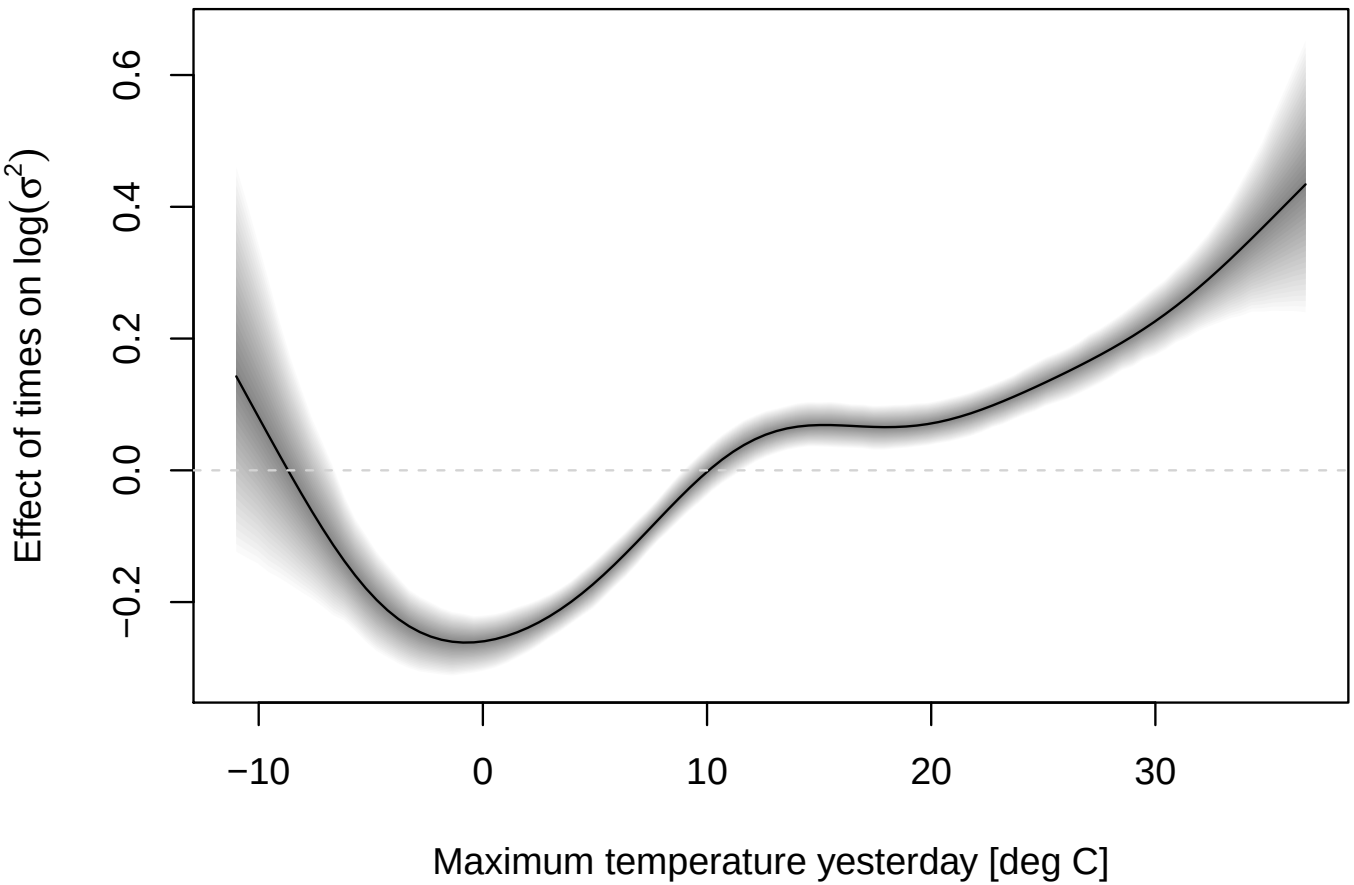
$$\mathrm{T} \sim N(\mu = f(\mathrm{T}_{t-1}),\ log(\sigma^2) = f(\mathrm{T}_{t-1})).$$



Introduction

Linz daily maximum temperature data (2000/01-2018/09)

$$\mathrm{T} \sim N(\mu = f(\mathrm{T}_{t-1}),\ log(\sigma^2) = f(\mathrm{T}_{t-1})).$$



Model specification

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

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

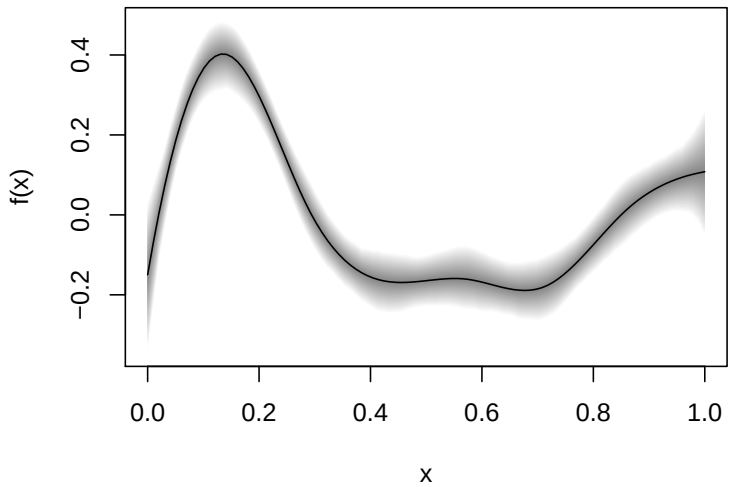
$j = 1, \ldots, J_k$ and $k = 1, \ldots, K$ and $h_k(\cdot)$ are known monotonic link functions.

Vector of function evaluations $\mathbf{f}_{jk} = (f_{jk}(\mathbf{x}_1; \boldsymbol{\beta}_{jk}), \ldots, f_{jk}(\mathbf{x}_n; \boldsymbol{\beta}_{jk}))^\top$

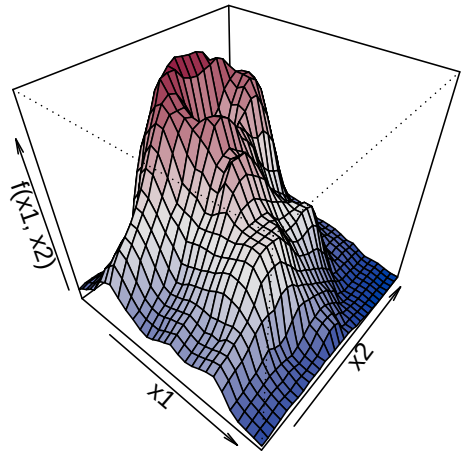
$$\mathbf{f}_{jk} = \begin{pmatrix} f_{jk}(\mathbf{x}_1; \boldsymbol{\beta}_{jk}) \\ \vdots \\ f_{jk}(\mathbf{x}_n; \boldsymbol{\beta}_{jk}) \end{pmatrix} = f_{jk}(\mathbf{X}_{jk}; \boldsymbol{\beta}_{jk}).$$

Model specification

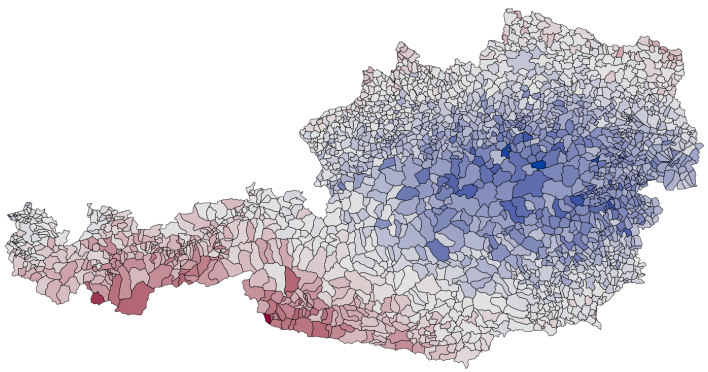
Nonlinear effects of continuous covariates



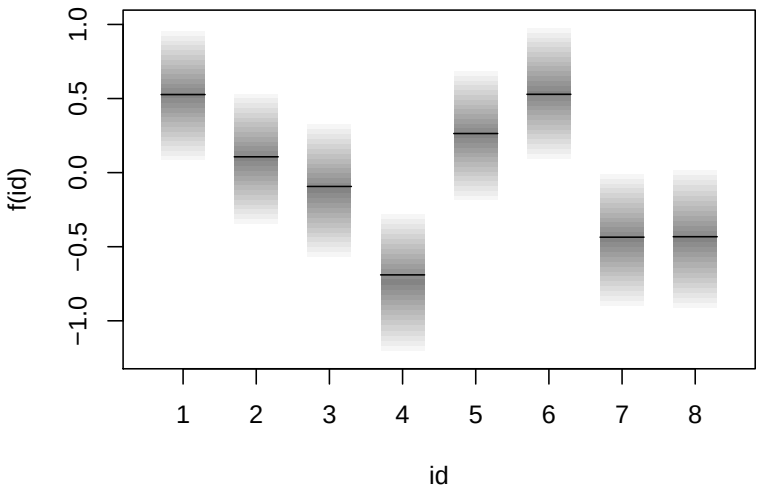
Two−dimensional surfaces



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

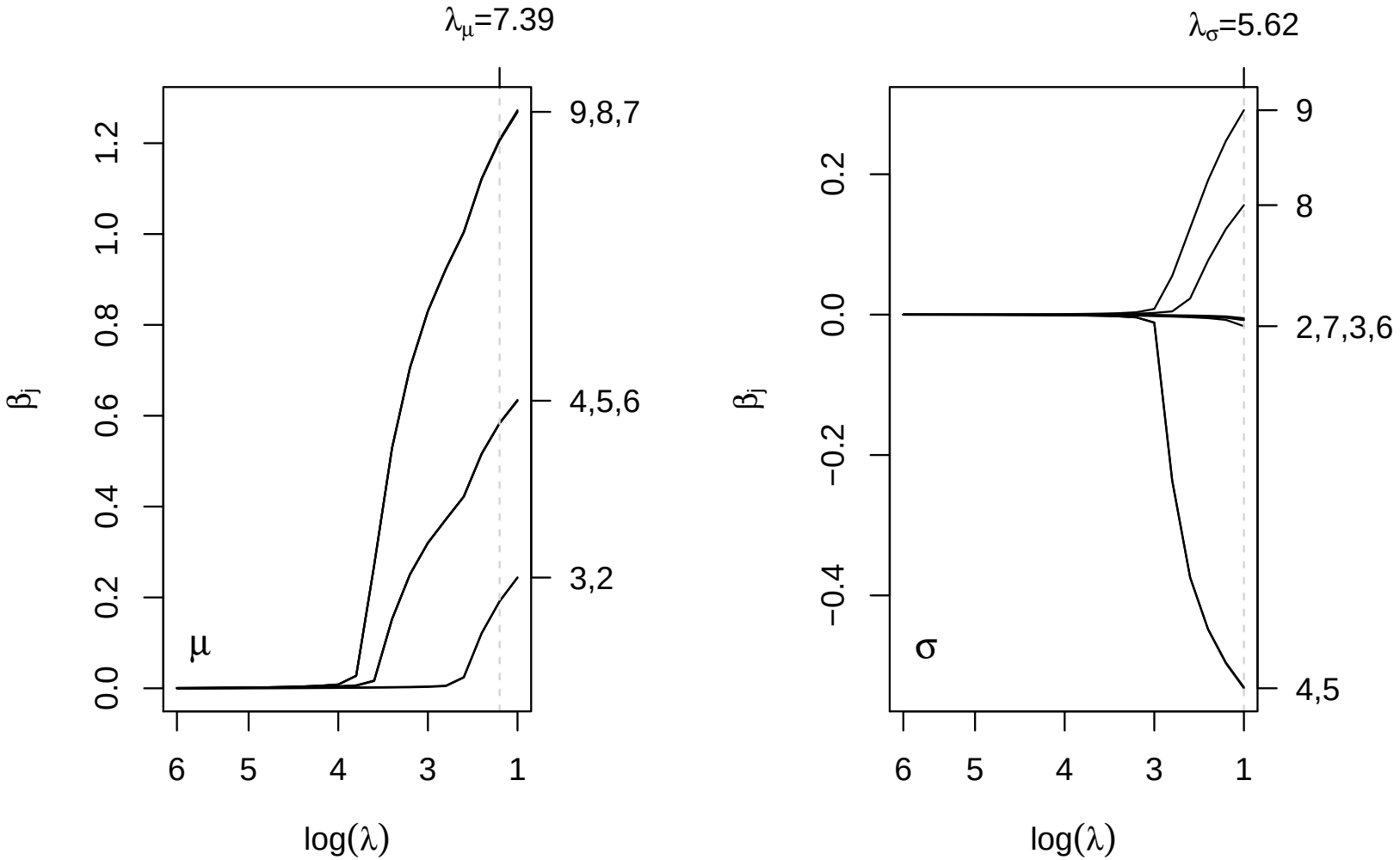


Random intercepts f(x) = f(id)



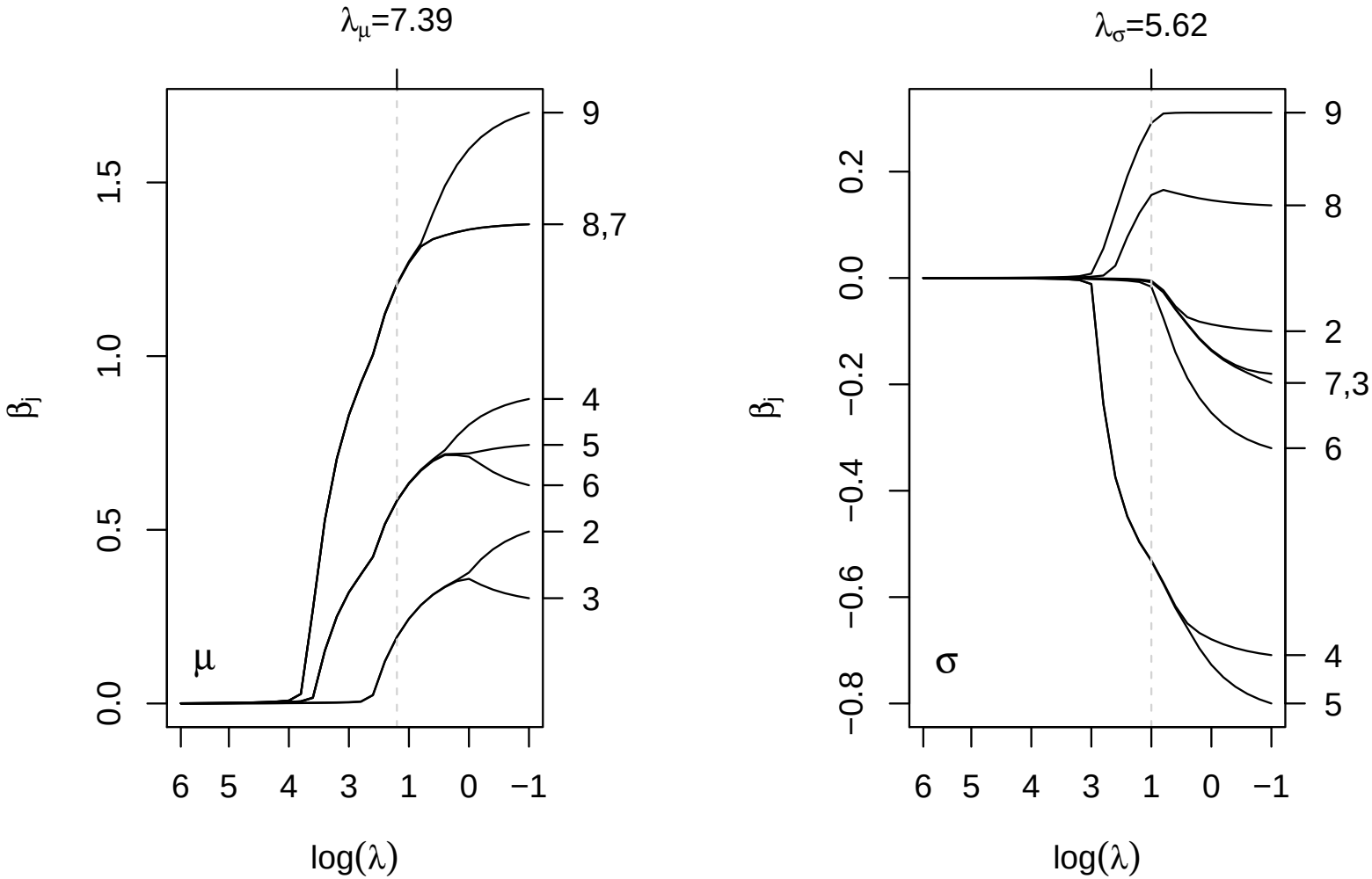
Model specification

Model terms $f_{jk}(\mathbf{x}; \boldsymbol{\beta}_{jk})$ with LASSO-type penalties $J_c(\boldsymbol{\beta}_{jk})$.



Model specification

Model terms $f_{jk}(\mathbf{x}; \beta_{jk})$ with LASSO-type penalties $J_f(\beta_{jk})$.



Model fitting

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

Estimation typically requires to evaluate

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

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

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

where $\boldsymbol{\tau} = (\boldsymbol{\tau}_1^\top, \ldots, \boldsymbol{\tau}_K^\top)^\top = (\boldsymbol{\tau}_{11}^\top, \ldots, \boldsymbol{\tau}_{J_1 1}^\top, \ldots, \boldsymbol{\tau}_{1K}^\top, \ldots, \boldsymbol{\tau}_{J_K K}^\top)^\top$ (frequentist, penalized log-likelihood).

Model fitting

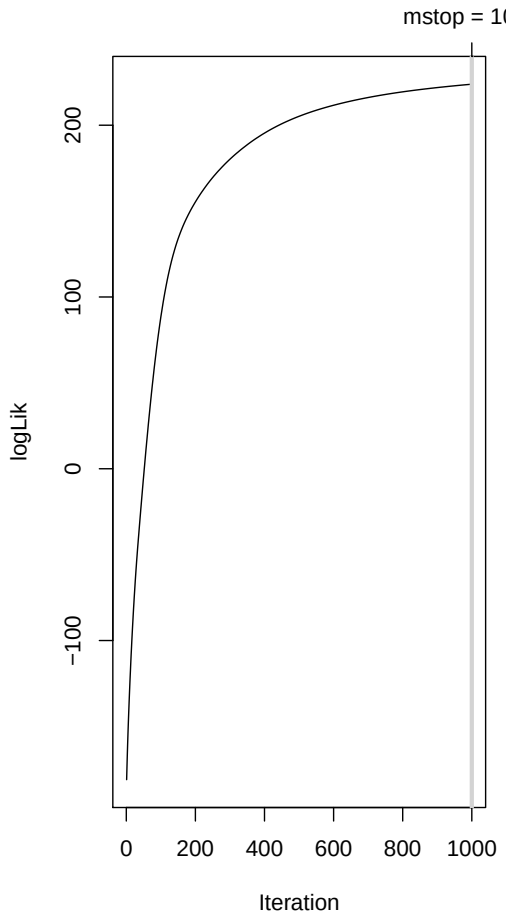
A simple generic algorithm for flexible regression models:

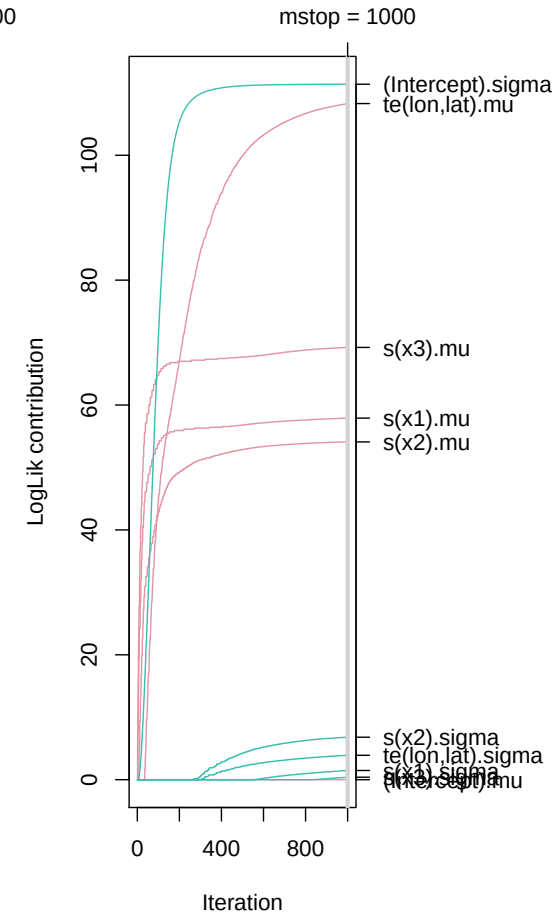
```
while(eps > ε & t < maxit) {
    for(k in 1:K) {
        for(j in 1:J[k]) {
            Compute η̃ = η_k − f_jk.
            Obtain new (β*_jk, τ*_jk)^⊤ = U_jk(X_jk, y, η̃, β^[t]_jk, τ^[t]_jk, α_jk).
            Update η_k = η̃ + f*_jk.
        }
    }
    t = t + 1
    Compute new eps.
}
```

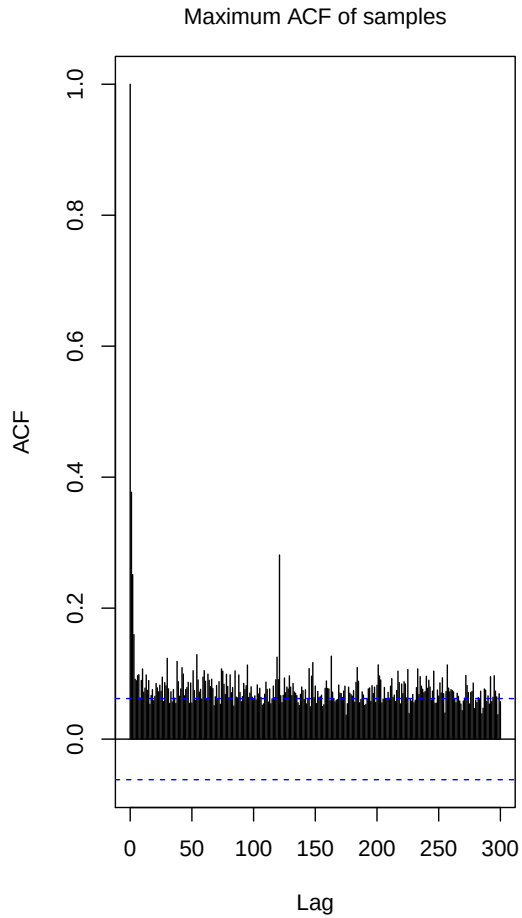
Functions $\mathtt{U}_{\mathtt{jk}}(\cdot)$ could either return updates from an optimizing algorithm or proposals from a MCMC sampler.

Model fitting

For complicated models use combination of algorithms, e.g., gradient boosting for finding starting values for MCMC.







R package *bamlss*

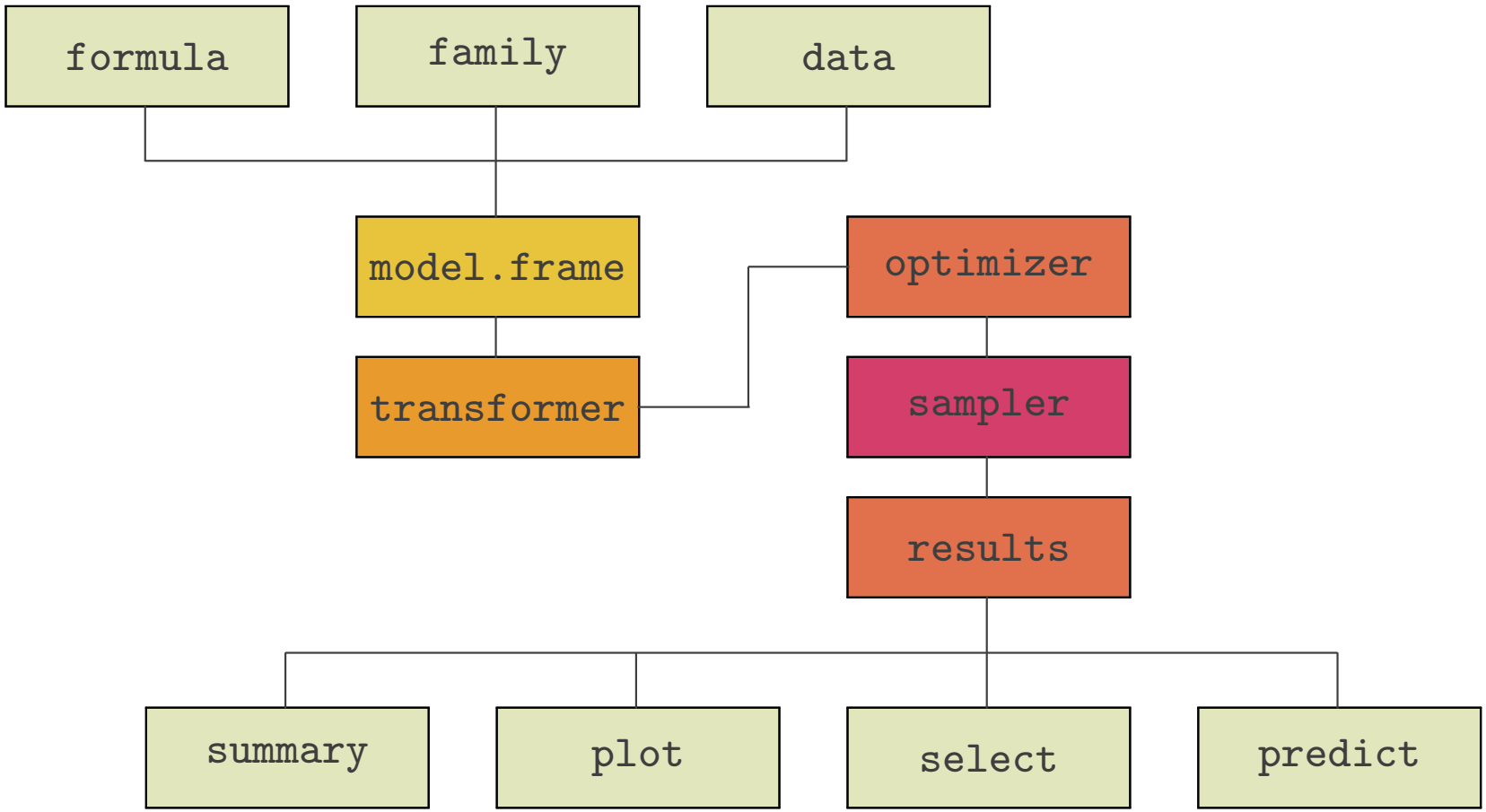
The package is available at

`https://CRAN.R-project.org/package=bamlss`

Development version, in R simply type

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

R package *bamlss*



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

R package *bamlss*

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

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

R package *bamlss*

The package makes heavy uses of **mgcv infrastructures** using `smooth.construct()`, however, optimizers and samplers may use special model terms, e.g., the LASSO constructor `la()`.

```
R> f <- list(
+   num ~ s(x1) + s(x2) + la(id),
+   sigma ~ s(x1) + s(x2) + la(id)
+ )
R> bf <- bamlss.frame(f, data = d, family = "gaussian")
R> names(bf)

[1] "call"        "model.frame" "y"             "formula"
[5] "terms"       "family"      "x"

R> names(bf$x$mu)

[1] "formula"           "fake.formula"     "terms"
[4] "model.matrix"      "smooth.construct"

R> names(bf$x$mu$smooth.construct)

[1] "s(x1)"  "s(x2)"  "la(id)"
```

R package *bamlss*

Work in progress . . .

Function	Distribution
`beta_bamlss()`	Beta distribution
`binomial_bamlss()`	Binomial distribution
`cnorm_bamlss()`	Censored normal distribution
`cox_bamlss()`	Continuous time Cox-model
`gaussian_bamlss()`	Gaussian distribution
`gamma_bamlss()`	Gamma distribution
`gpareto_bamlss()`	Generalized Pareto distribution
`jm_bamlss()`	Continuous time joint-model
`multinomial_bamlss()`	Multinomial distribution
`mvn_bamlss()`	Multivariate normal distribution
`poisson_bamlss()`	Poisson distribution
...	

New families only require density, distribution, random number generator, quantile, score and hess functions. Wrapper for R package *gamlss* families.

R package *bamlss*

Wrapper function:

```
R> f <- list(y ~ la(id,fuse=2), sigma ~ la(id,fuse=1))
R> b <- bamlss(f, family = "gaussian", sampler = FALSE,
+    optimizer = lasso, criterion = "BIC", multiple = TRUE)
```

Standard extractor and plotting functions:

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

R package *bamlss*

Example: model fitting functions.

```
bfit(x, y, family, start = NULL, weights = NULL, offset = NULL,
  update = "iwls", criterion = c("AICc", "BIC", "AIC"), ...)
```

```
boost(x, y, family, weights = NULL, offset = NULL,
  nu = 0.1, df = 4, maxit = 400, ...)
```

```
GMCMC(x, y, family, start = NULL, weights = NULL, offset = NULL,
  n.iter = 1200, burnin = 200, thin = 1, ...)
```

R package *bamlss*

Example: updating functions.

```
bfit_iwls(x, family, y, eta, id, weights, criterion, ...)
```

```
boost_fit(x, y, nu, hatmatrix = TRUE, weights = NULL, ...)
```

```
GMCMC_iwls(family, theta, id, eta, y, data,
  weights = NULL, offset = NULL, ...)
```

```
GMCMC_slice(family, theta, id, eta, y, data, ...)
```

Forecasting Lightning

Objective: Develop a NWP postprocessing for probabilistic forecasting of lightning counts in the Eastern Alps.

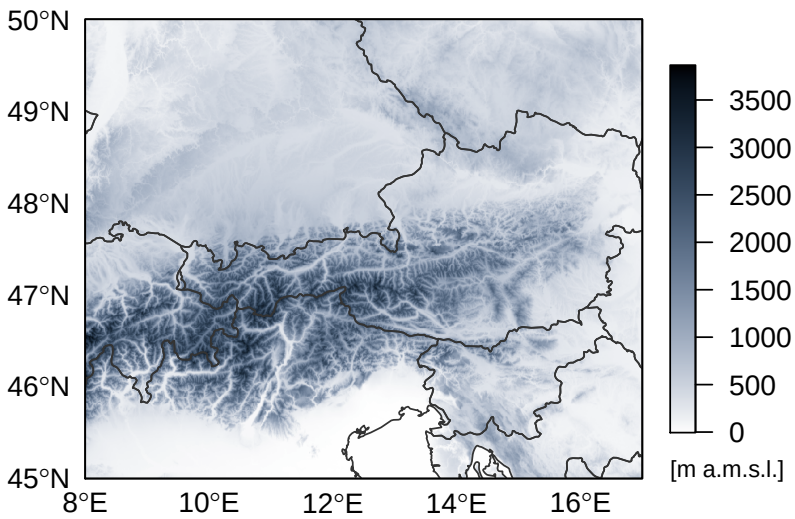
Requirements:

- Handle **nonlinear** relationships between the response and covariates.
- **Select** objectively important exploratory variables.
- Provide **inference** of scores and predictions.

Data

ALDIS lightning counts:

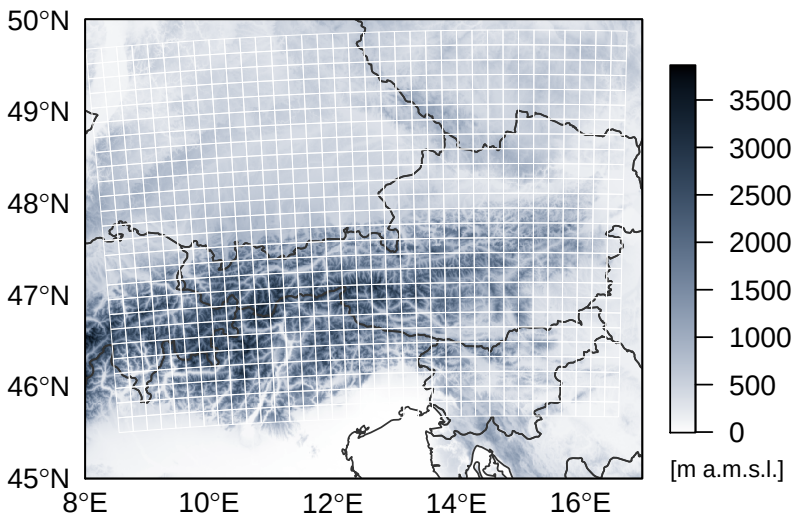
- Summer: May–August.
- Afternoons: 12–18 UTC.
- Gridded on 18×18 km^2.
- 2010–2017.



Data

ALDIS lightning counts:

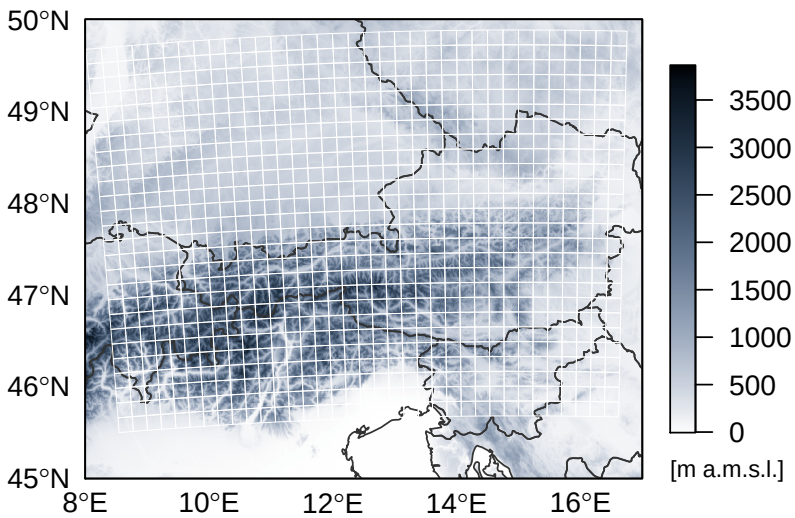
- Summer: May–August.
- Afternoons: 12–18 UTC.
- Gridded on 18×18 km^2.
- 2010–2017.



Data

ALDIS lightning counts:

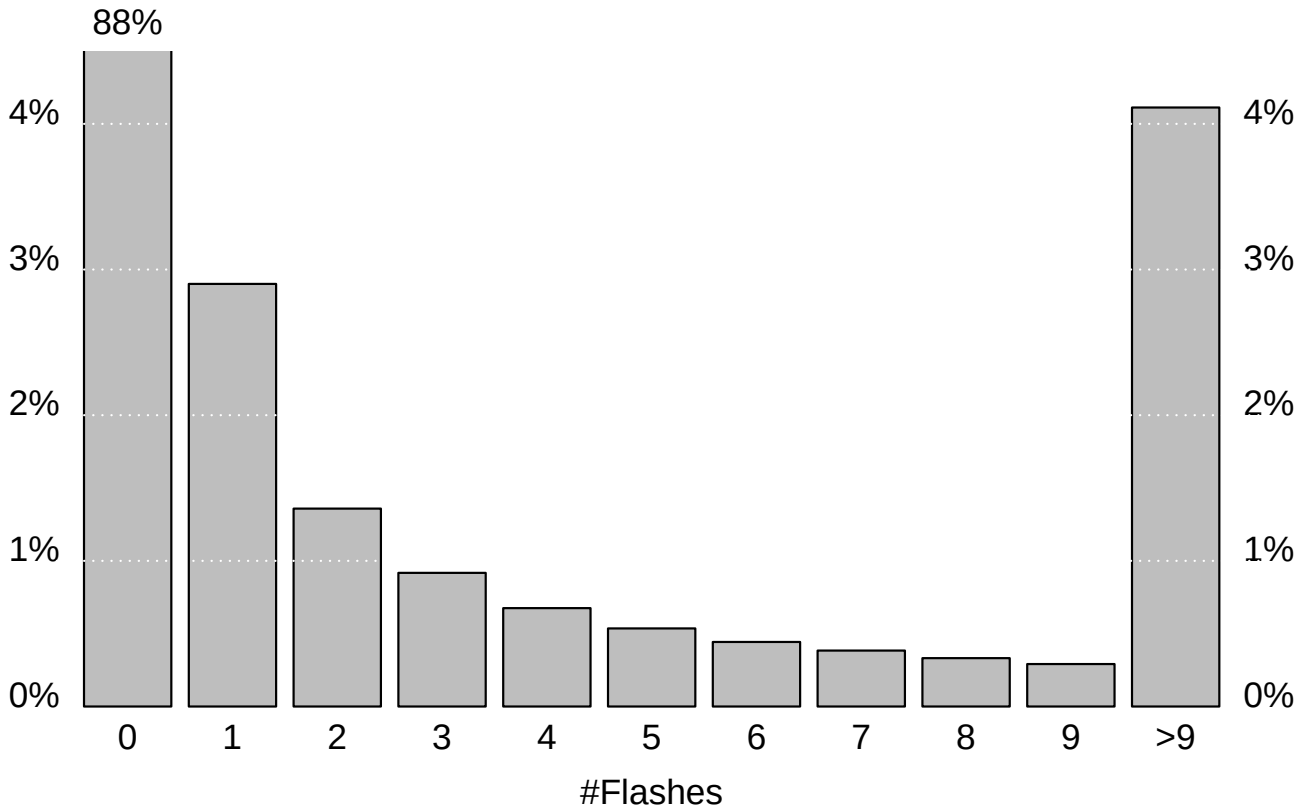
- Summer: May–August.
- Afternoons: 12–18 UTC.
- Gridded on 18×18 km^2.
- 2010–2017.



ECMWF ENS:

- Forecast horizons: 1–5 days.
- 2016–2017.
- Convective precipitation, CAPE, temperature, relative humidity, vertical velocity, radiation, heat fluxes, ...
- Median and interquartile range.

Lightning Counts



Methods

- Excess zeros and overdisperion in count data:
  Hurdle approach.

- Nonlinear relationships:
  Additive predictors.

- Selection of terms:
  Gradient boosting with stability selection.

- Inference:
  Markov chain Monte Carlo simulation.

Hurdle approach

Form an **intensity model** of lightning events by combining

- **Binary hurdle part:**
  Logit binomial with parameter π.

  Serves as isolated model for the **occurrence** of lightning events.

- **Truncated count part:**
  Zero-truncated negative binomial with location parameter $\mu > 0$ and dispersion parameter $\theta > 0$.

Additive predictors

$$\text{logit}(\pi) = \underbrace{\beta_0 + f_1(\texttt{doy}) + f_2(\texttt{lon}, \texttt{lat})}_{\text{climatology}} + f_3(\texttt{x}_3) + \cdots + f_p(\texttt{x}_p).$$

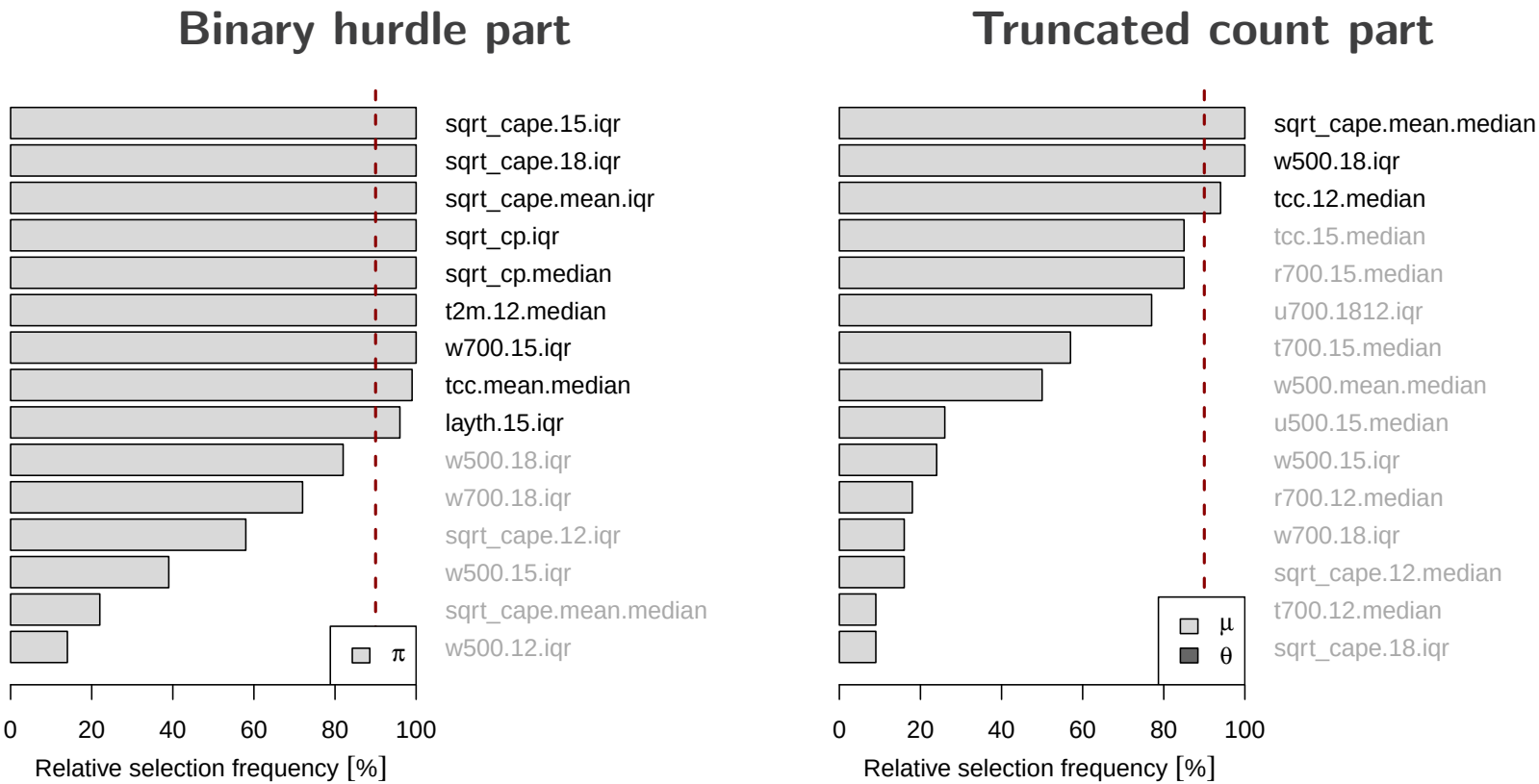
$$\log(\mu) = \ldots$$

$$\log(\theta) = \ldots$$

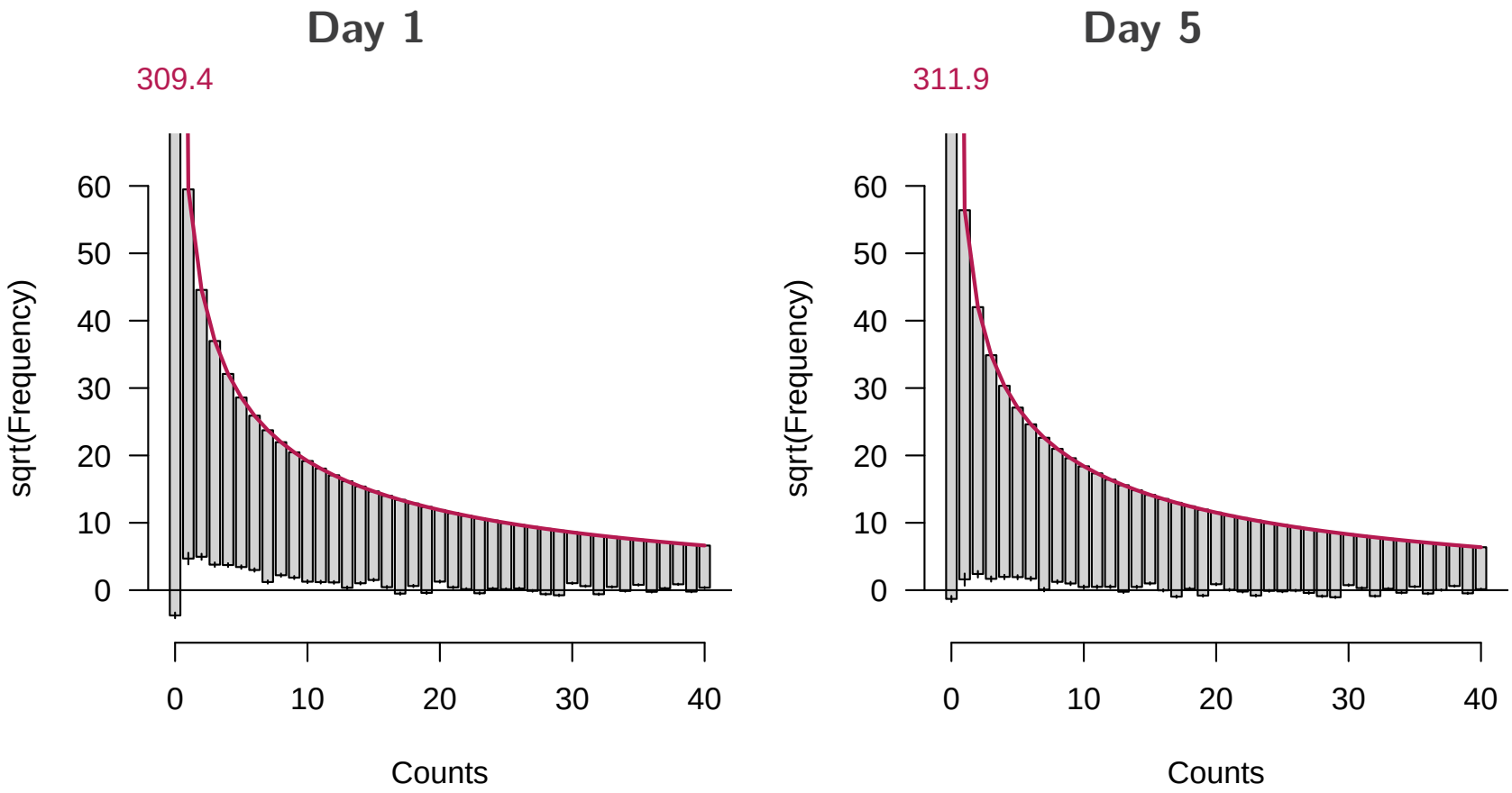
- π, μ, θ: Distributional parameters.
- $f_i(\cdot)$: Potentially non-linear functions.
- `doy`: Day of the year.
- `lon`, `lat`: Longitude, latitude.
- $\texttt{x}_i$: ECMWF-ENS covariate.
- p: Number of potential terms (here: $p \approx 200$).

Stability Selection

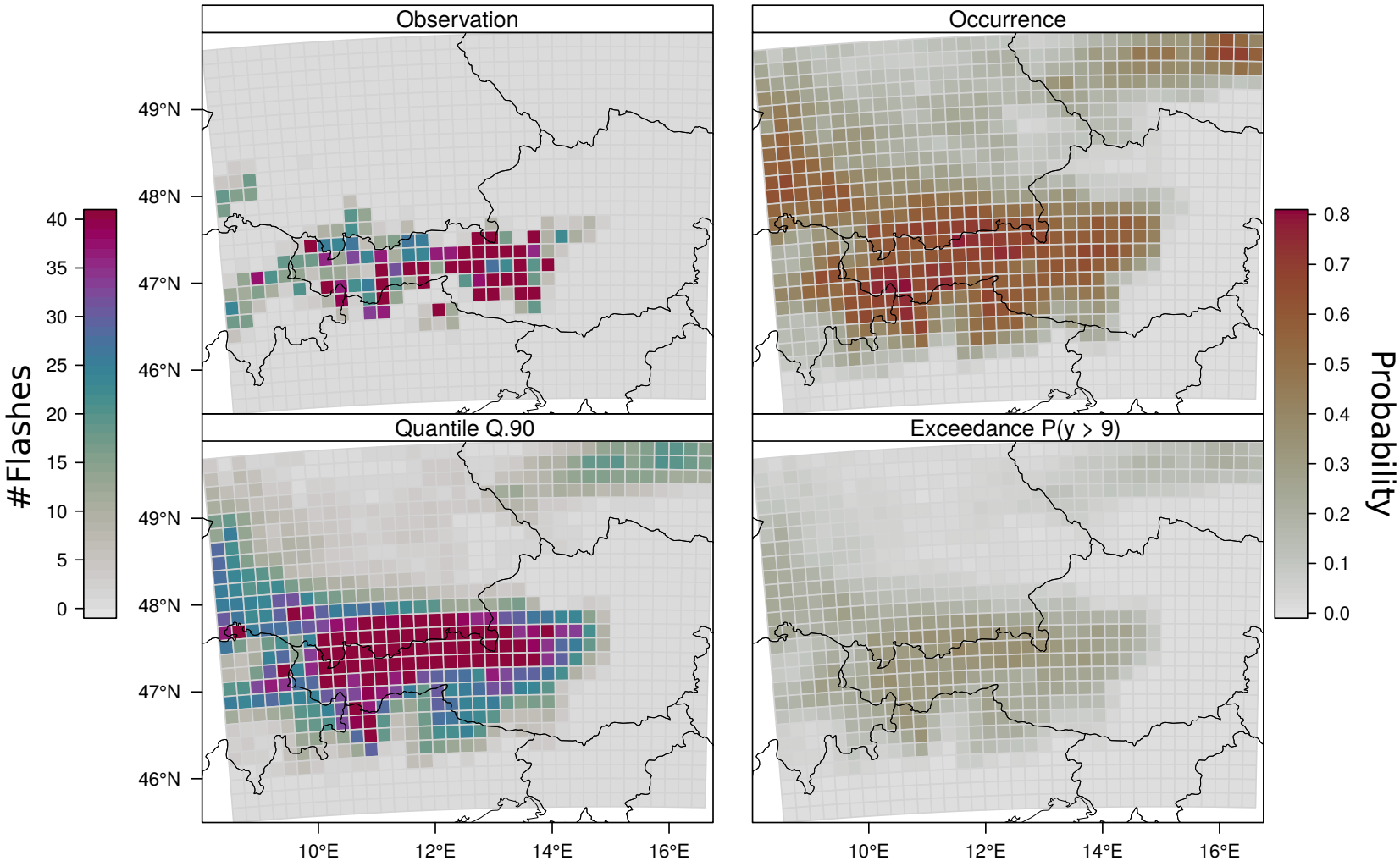
Function `stabsel()` in *bamlss*.



Predictive Performance



Predictions



References

- A. Groll, et al. *LASSO-Type Penalization in the Framework of Generalized Additive Models for Location Scale and Shape*. Working Papers, Faculty of Economics and Statistics, University of Innsbruck, 2018. https://EconPapers.repec.org/RePEc:inn:wpaper:2018-16
- M. Köhler, et al. *Nonlinear Association Structures in Flexible Bayesian Additive Joint Models*. STAT MED, 2018. doi:10.1002/sim.7967.
- T. Simon, et al. *Lightning Prediciton Using Model Output Statistics*. Working Papers, Faculty of Economics and Statistics, University of Innsbruck, 2018. https://EconPapers.repec.org/RePEc:inn:wpaper:2018-14.
- N. Umlauf, et al. *bamlss: Bayesian Additive Models for Location Scale and Shape (and Beyond)*. R package version 1.0-0, 2018. http://CRAN.R-project.org/package=bamlss.
- N. Umlauf, et al. *BAMLSS: Bayesian Additive Models for Location, Scale and Shape (and Beyond)*. J COMPUT GRAPH STAT, 2017. doi:10.1080/10618600.2017.1407325.
- A. Zeileis, et al. *Visualizing Count Data Regressions Using Rootograms*. AM STAT, 2016. doi:10.1080/00031305.2016.1173590.

Thank you for your attention!

Nikolaus Umlauf

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