Scalable Distributional Regression

Nikolaus Umlauf

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

Bayes@Austria WU Wien, 2020-11-28.

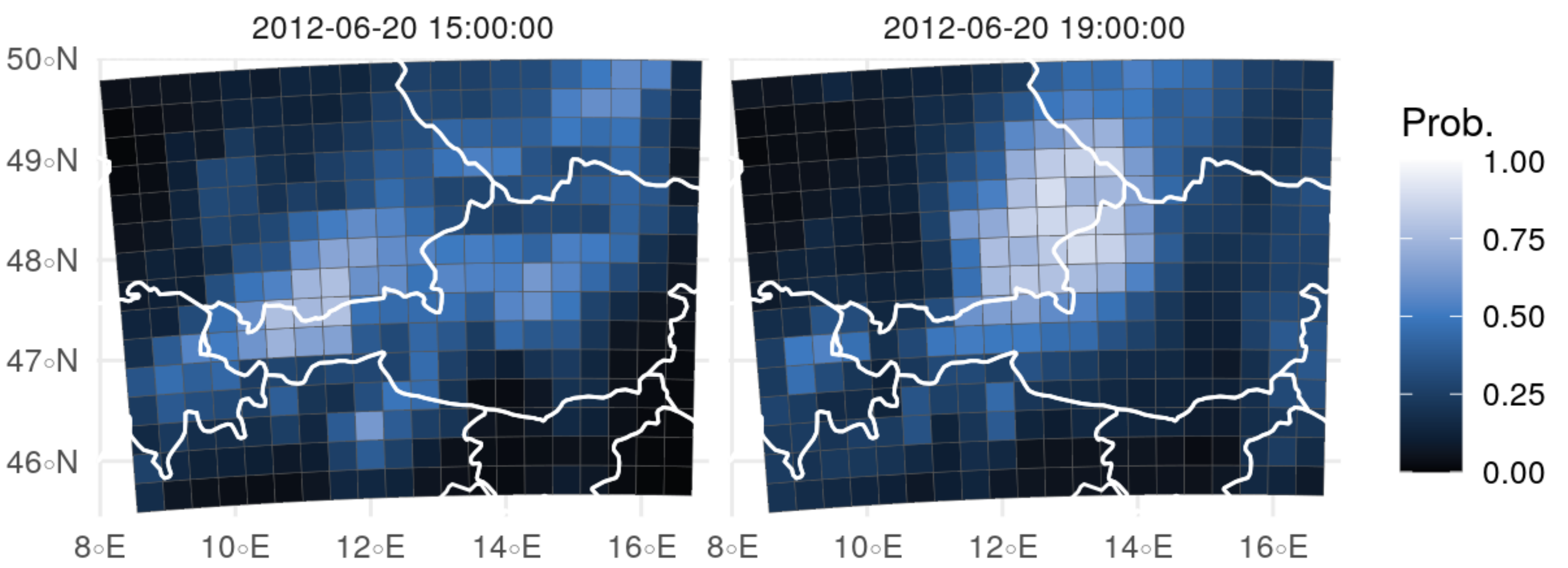
foto-webcam.eu, Innsbruck Seegrube, 2017-07-29

Forecasting Lightning

Goal: Forecast lightning by statistical post-processing of numerical weather prediction (NWP) output.

(a) **Occurence**: Is there any lightning? (Binary)

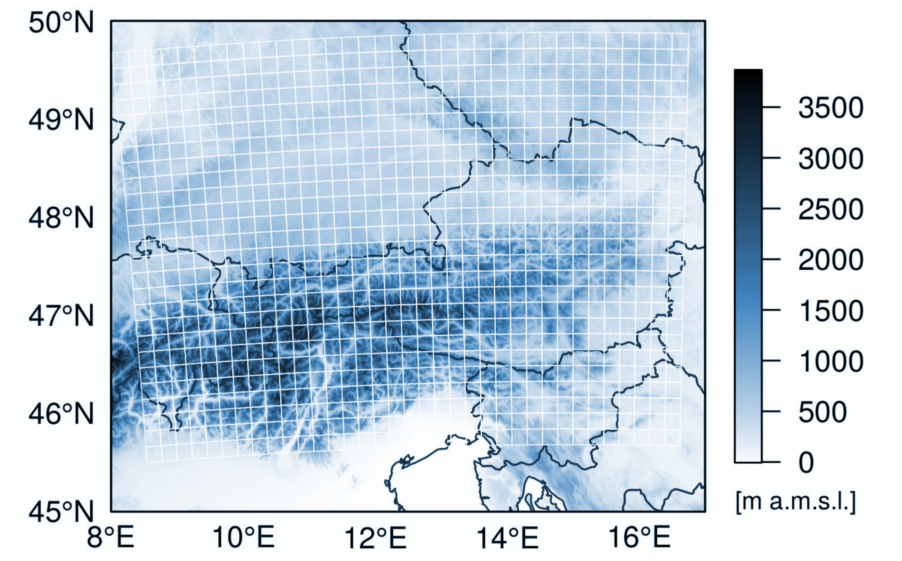
(b) **Intensity**: If there is any lightning, how many? (Counts > 0)



Data

ALDIS lightning counts:

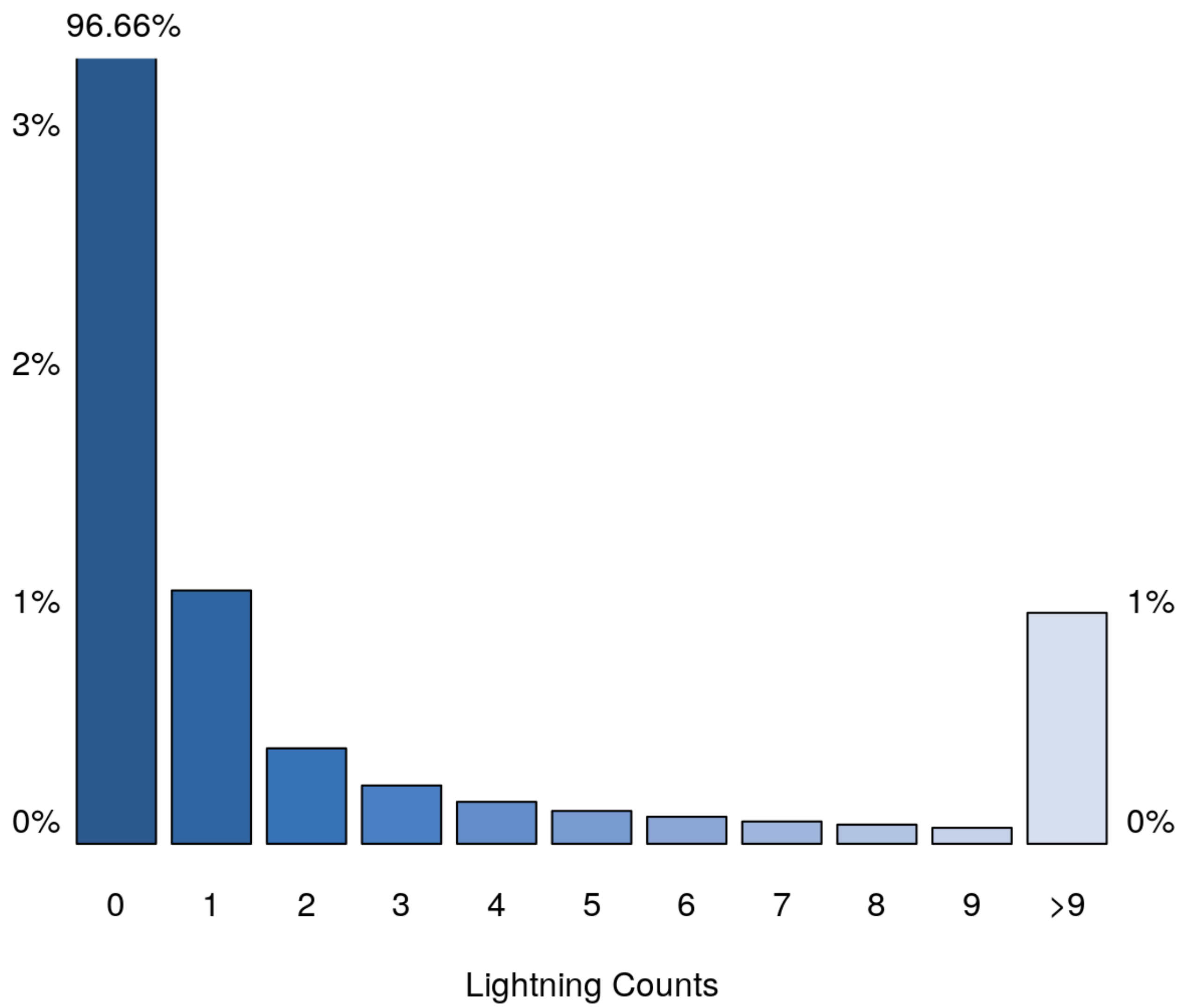
- Summer: May-August.
- Afternoons: 12-18 UTC.
- #Obs. ~ 8M.
- Gridded on 18 × 18 km^2.
- 2010-2017.



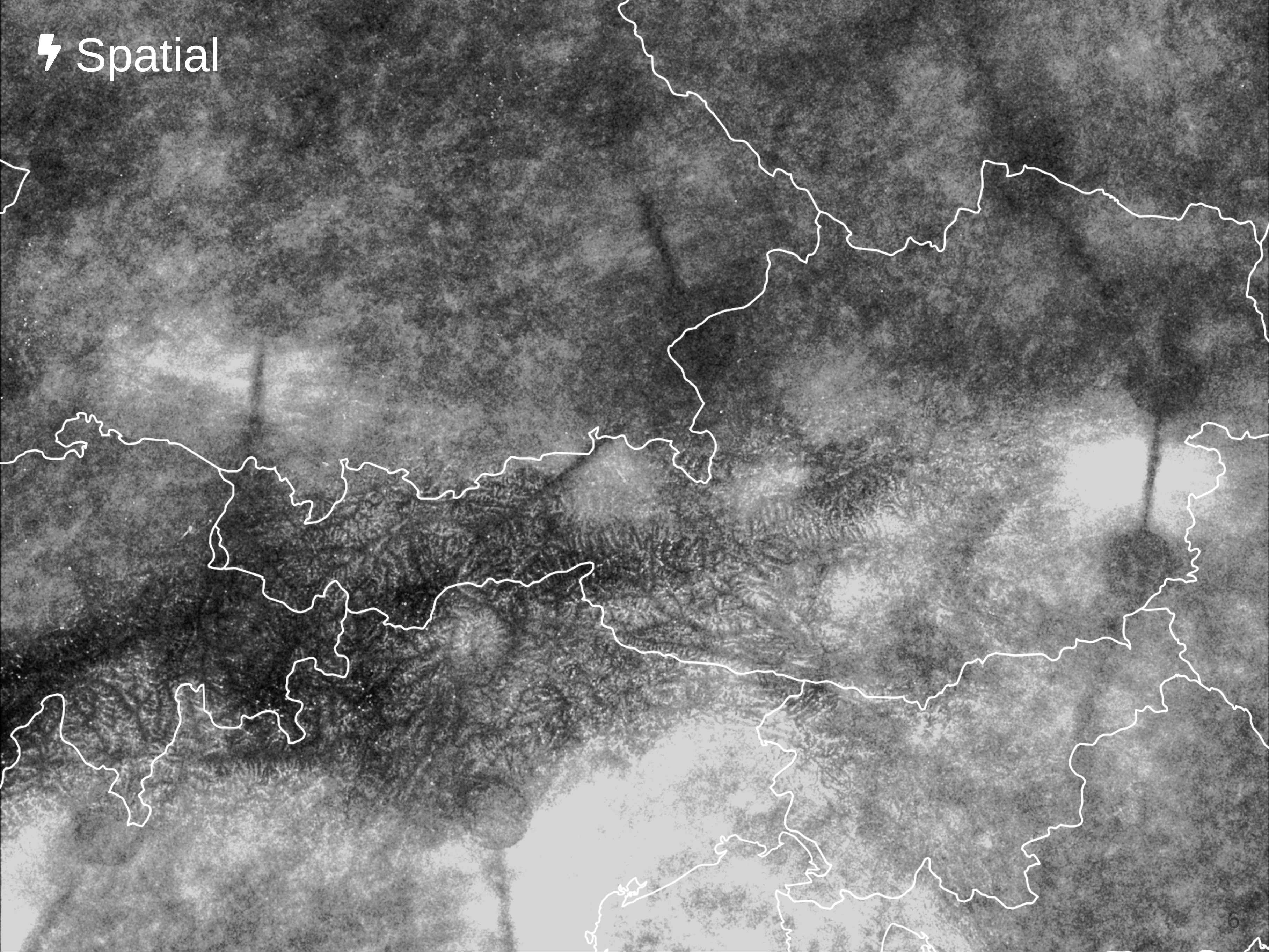
ECMWF ensemble forecasts:

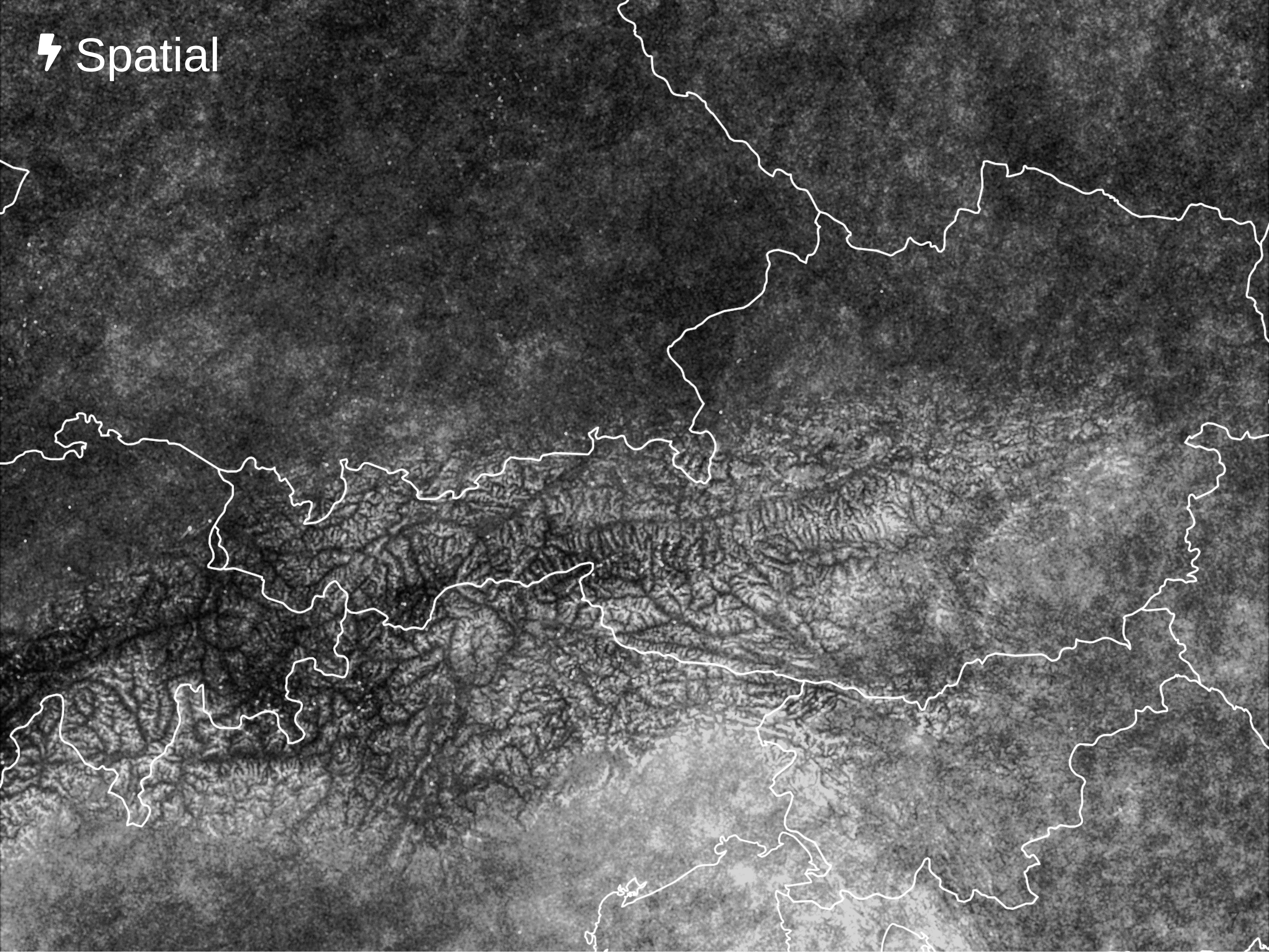
- Forecast horizons: 1-5 days.
- 2010-2017.
- NWP outputs: Convective precipitation, CAPE, temperature, relative humidity, vertical velocity, radiation, heat fluxes, ...
- Median and interquartile range.

Lightning Counts



Spatial




Spatial

Forecasting Lightning

Model requirements:

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

Software requirements:

- Very **flexible** regression model.
- Very **large** dataset.
- Computationally **intensive**.
- Implementation is **not** straightforward.

Lego Toolbox

Hence: Flexible regression framework for Bayesian additive models for location, scale, and shape (BAMLSS).

Software: *R* package **bamlss**. Modular design supports easy development.

Software Design

Input — Data, distribution, regression.

Pre-processing — Model frame, transformations.

Estimation — Optimizer and/or sampler functions.

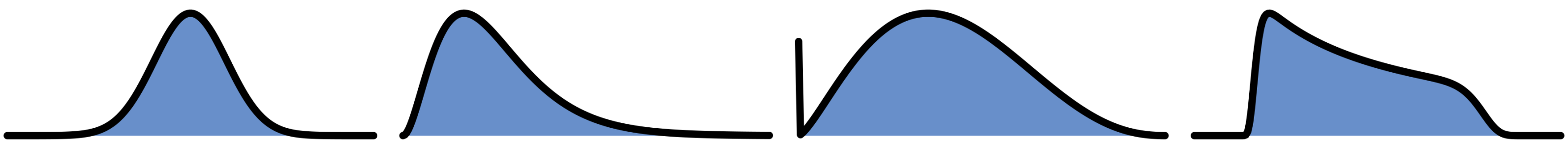
Post-processing — Sampling statistics & results.

Output — Prediction, model selection, visualization, ...

Model Specification

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

$$y \sim \mathcal{D}\left(\theta_1(\mathbf{x}; \boldsymbol{\beta}_1), \ \ldots, \ \theta_K(\mathbf{x}; \boldsymbol{\beta}_K)\right),$$



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

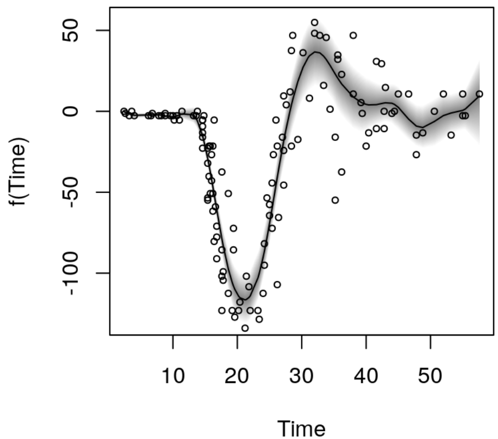
Each parameter is linked to a structured additive predictor

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

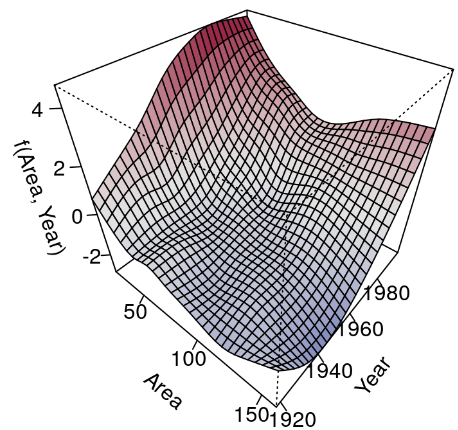
- $j = 1, \ldots, J_k$ and $k = 1, \ldots, K$.
- $h_k(\cdot)$: Link functions for each distribution parameter.
- $f_{jk}(\cdot)$: Model terms of one or more variables.

Model Terms $f_{jk}(\cdot)$

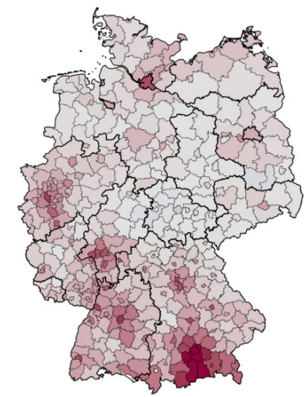
Nonlinear Effects



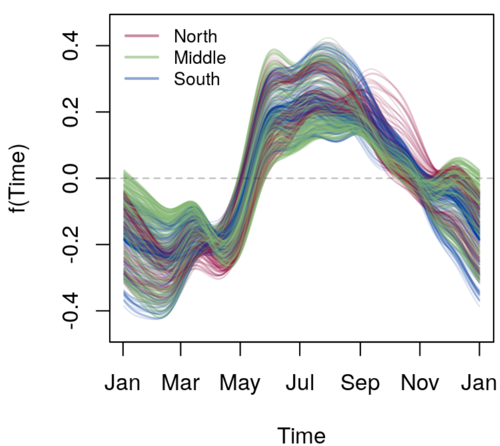
Interaction Surfaces



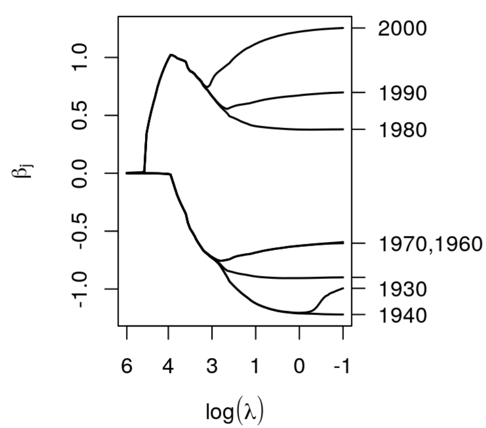
Spatially Correlated Effects f(x) = f(s)



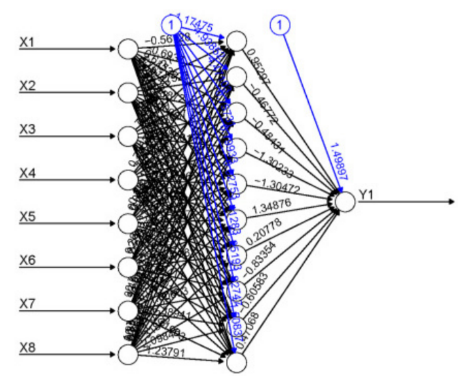
Space-Time Varying Effects



LASSO & Factor Clustering



Neural Networks



Model Fitting

The main building block is $d_y(\mathbf{y}|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$.

Estimation typically requires to evaluate the log-likelihood

$$\ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{n} \log\, d_y(y_i; \theta_1(\mathbf{x}_i; \boldsymbol{\beta}_1),\ \ldots\ ,\ \theta_K(\mathbf{x}_i; \boldsymbol{\beta}_K)),$$

with $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_K)$.

The log-posterior (frequentist penalized log-likelihood)

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

where $p_{jk}(\cdot)$ are priors, $\boldsymbol{\tau}_{jk}$ (smoothing) variances and $\boldsymbol{\alpha}_{jk}$ fixed hyper parameters.

Priors $p_{jk}(\cdot)$

For simple linear effects $\mathbf{X}_{jk}\boldsymbol{\beta}_{jk}$: $p_{jk}(\boldsymbol{\beta}_{jk}) \propto \text{const.}$

For the smooth terms:

$$p_{jk}(\boldsymbol{\beta}_{jk}; \boldsymbol{\tau}_{jk}, \boldsymbol{\alpha}_{jk}) \propto d_{\boldsymbol{\beta}_{jk}}(\boldsymbol{\beta}_{jk} | \, \boldsymbol{\tau}_{jk}; \boldsymbol{\alpha}_{\boldsymbol{\beta}_{jk}}) \cdot d_{\boldsymbol{\tau}_{jk}}(\boldsymbol{\tau}_{jk} | \, \boldsymbol{\alpha}_{\boldsymbol{\tau}_{jk}}).$$

Using a basis function approach a common choice is

$$d_{\boldsymbol{\beta}_{jk}}(\boldsymbol{\beta}_{jk} | \, \boldsymbol{\tau}_{jk}, \boldsymbol{\alpha}_{\boldsymbol{\beta}_{jk}}) \propto |\mathbf{P}_{jk}(\boldsymbol{\tau}_{jk})|^{\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{\beta}_{jk}^{\top}\mathbf{P}_{jk}(\boldsymbol{\tau}_{jk})\boldsymbol{\beta}_{jk}\right).$$

Precision matrix $\mathbf{P}_{jk}(\boldsymbol{\tau}_{jk})$ derived from prespecified penalty matrices $\boldsymbol{\alpha}_{\boldsymbol{\beta}_{jk}} = \{\mathbf{K}_{1jk}, \ldots, \mathbf{K}_{Ljk}\}$.

The variances parameters $\boldsymbol{\tau}_{jk}$ are equivalent to the inverse smoothing parameters in a frequentist approach.

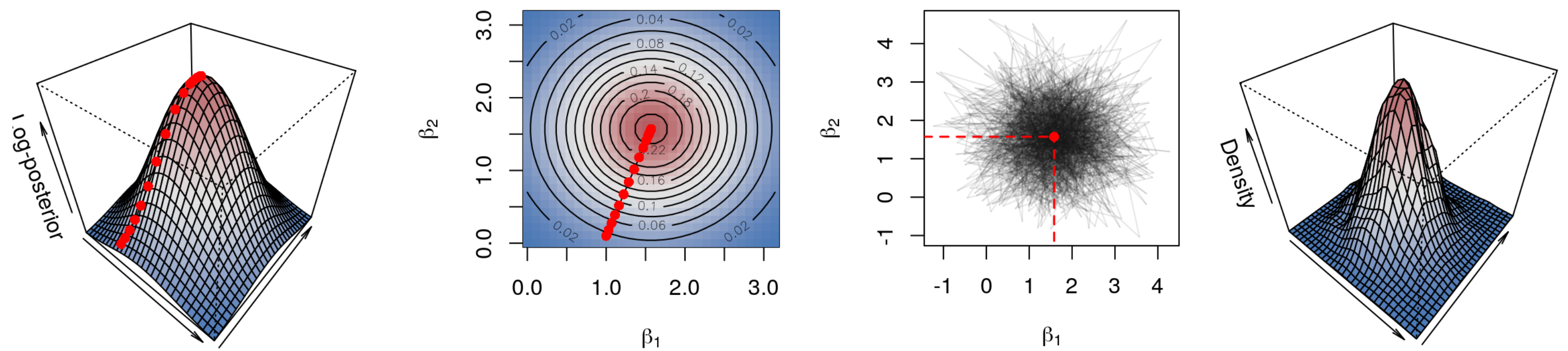
Estimation

Bayesian point estimates of parameters are obtained by:

1. Maximization of the log-posterior for posterior mode estimation.
2. Solving high dimensional integrals, e.g., for posterior mean or median estimation.

Problems 1 and 2 are commonly solved by computer intensive iterative algorithms of the following type:

$$(\boldsymbol{\beta}^{[t+1]}, \boldsymbol{\tau}^{[t+1]}) = \mathtt{U}(\boldsymbol{\beta}^{[t]}, \boldsymbol{\tau}^{[t]}; \mathbf{y}, \mathbf{X}, \boldsymbol{\alpha}).$$



Updating

Example: MCMC updating functions $\mathtt{U}_{jk}(\cdot)$.

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

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

  Metropolis-Hastings acceptance probability

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

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

Scalable Distributional Regression

Lightning data:

- Lightning dataset includes >100 variables from ECMWF ensemble forecasts.
- #Obs. ~ 8M.

Challenges:

- Select only relevant variables.
- Algorithms for very large datasets in distributional regression?
- The aim is to run the analysis for all of Europe!

➔ **Efficient algorithm** with a **small memory footprint**?!

Scalable Distributional Regression

Consider the following updating scheme

$$\boldsymbol{\beta}_k^{[t+1]} = \mathtt{U}_k(\boldsymbol{\beta}_k^{[t]};\,\cdot\,) = \boldsymbol{\beta}_k^{[t]} - \mathbf{H}_{kk}\left(\boldsymbol{\beta}_k^{[t]}\right)^{-1}\mathbf{s}\left(\boldsymbol{\beta}_k^{[t]}\right).$$

Assuming model terms that can be written as a matrix product of a design matrix and coefficients we obtain an iteratively weighted least squares scheme given by

$$\begin{aligned}\boldsymbol{\beta}_{jk}^{[t+1]} &= \mathtt{U}_{jk}(\boldsymbol{\beta}_{jk}^{[t]};\,\cdot\,) \\ &= (\mathbf{X}_{jk}^\top \mathbf{W}_{kk}\mathbf{X}_{jk} + \mathbf{G}_{jk}(\boldsymbol{\tau}_{jk}))^{-1}\mathbf{X}_{jk}^\top \mathbf{W}_{kk}(\mathbf{z}_k - \boldsymbol{\eta}_{k,-j}^{[t+1]}),\end{aligned}$$

with working observations $\mathbf{z}_k = \boldsymbol{\eta}_k^{[t]} + \mathbf{W}_{kk}^{-1\,[t]}\mathbf{u}_k^{[t]}$, working weights $\mathbf{W}_{kk}^{-1\,[t]}$ and score vector $\mathbf{u}_k^{[t]}$.

Scalable Distributional Regression

Instead of using all observations of the data, we only use a randomly chosen **subset** denoted by the subindex $[\mathbf{s}]$ in one updating step

$$\boldsymbol{\beta}_{jk}^{[t+1]} = \nu \cdot (\mathbf{X}_{[\mathbf{s}],jk}^{\top} \mathbf{W}_{[\mathbf{s}],kk} \mathbf{X}_{[\mathbf{s}],jk} + \mathbf{G}_{jk}(\boldsymbol{\tau}_{jk}))^{-1} \mathbf{X}_{[\mathbf{s}],jk}^{\top} \mathbf{W}_{[\mathbf{s}],kk} (\mathbf{z}_{[\mathbf{s}],k} - \boldsymbol{\eta}_{[\mathbf{s}],k,-j}^{[t+1]}) + \\ (1 - \nu) \cdot \boldsymbol{\beta}_{jk}^{[t]},$$

where ν is a weight parameter which specifies how much the parameters at iteration $t + 1$ are influenced by parameters of the previous iteration t.

Use **flat file** format for each $\mathbf{X}_{jk}$, i.e., only batch $[s]$ is in memory. This way, we can estimate models with **really** large datasets.

Scalable Distributional Regression

Mimics a second order **stochastic gradient descent** (SGD) algorithm

$$\boldsymbol{\beta}_{jk}^{[t+1]} = \boldsymbol{\beta}_{jk}^{[t]} + \nu \cdot (\boldsymbol{\beta}_{jk,[\mathbf{s}]} - \boldsymbol{\beta}_{jk}^{[t]}) = \boldsymbol{\beta}_{jk}^{[t]} + \nu \cdot \boldsymbol{\delta}_{jk}^{[t]},$$

and $\boldsymbol{\delta}_{jk}^{[t]}$ is composed from first and second order derivative information with

$$\begin{aligned}\boldsymbol{\delta}_{jk}^{[t]} &= \boldsymbol{\beta}_{jk,[\mathbf{s}]} - \boldsymbol{\beta}_{jk}^{[t]} \\ &= \left[\boldsymbol{\beta}_{jk}^{[t]} - \mathbf{H}_{[\mathbf{s}],kk}\left(\boldsymbol{\beta}_{jk}^{[t]}\right)^{-1} \mathbf{s}_{[\mathbf{s}]}\left(\boldsymbol{\beta}_{jk}^{[t]}\right)\right] - \boldsymbol{\beta}_{jk}^{[t]} \\ &= -\mathbf{H}_{[\mathbf{s}],kk}\left(\boldsymbol{\beta}_{jk}^{[t]}\right)^{-1} \mathbf{s}_{[\mathbf{s}]}\left(\boldsymbol{\beta}_{jk}^{[t]}\right)\end{aligned}$$

Hence, the updating step length is adaptive.

Scalable Distributional Regression

Overfitting:

The idea is to select $\boldsymbol{\tau}_{jk}$ using a stepwise algorithm which is based on an **"out-of-sample" criterion**, i.e., the criterion $C(\cdot)$ is evaluated on another batch denoted by $[\tilde{\mathbf{s}}]$, $C_{[\tilde{\mathbf{s}}]}(\cdot)$ respectively, i.e.

$$\tau_{ljk}^{[t+1]} \leftarrow \underset{\tau_{ljk}^{\star} \in \mathcal{I}_{ljk}}{\arg\min}\, C_{[\tilde{\mathbf{s}}]}(U_{jk}(\boldsymbol{\beta}_{jk}^{[t]}, \tau_{ljk}^{\star};\ \cdot)),$$

where $\mathcal{I}_{ljk}$ is a search interval for $\tau_{ljk}^{[t+1]}$, e.g.,

$$\mathcal{I}_{ljk} = [\tau_{ljk}^{[t]} \cdot 10^{-1}, \tau_{ljk}^{[t]} \cdot 10].$$

Scalable Distributional Regression

Some interesting features:

1. Set, e.g., $\nu = 0.1$, convergence after visiting m batches $[\mathbf{s}]$.
2. Only update if **"out-of-sample" log-likelihood** is **increased**.
3. **Boosting** for **variable selection**: Update only $f_{jk}(\cdot)$ with greatest contribution in "out-of-sample" log-likelihood.
4. **Bagging**: If $\nu = 1$, each update is so to say a **"sample"**. Convergence similar to MCMC algorithms, i.e., if $\boldsymbol{\beta}_{jk}^{[t+1]}$ start fluctuating around a certain level.
5. **Slice sample** τ_{ljk} under $C_{[\tilde{\mathbf{s}}]}(\cdot)$, **much faster**!

Neural Network Terms $f_{jk}(\cdot)$

Motivation:

- Lightning model.
- Complex **nonlinearities** in the atmosphere?
- Neural networks (NN) are **universal** function approximators.

Problems:

- Estimation is difficult and can involve **thousands** of parameters.
- Fully **Bayesian** inference?

Solution:

- Use NNs based on **random** (inner) weights.
- Recently, detailed description on weight sampling available.
- Combine with **LASSO shrinkage**.

Lego in Action

Count distribution: Discrete generalized Pareto $\mathcal{DGP}(\xi, \sigma)$.

Regression: Smooth terms for NWP output variables & NN.

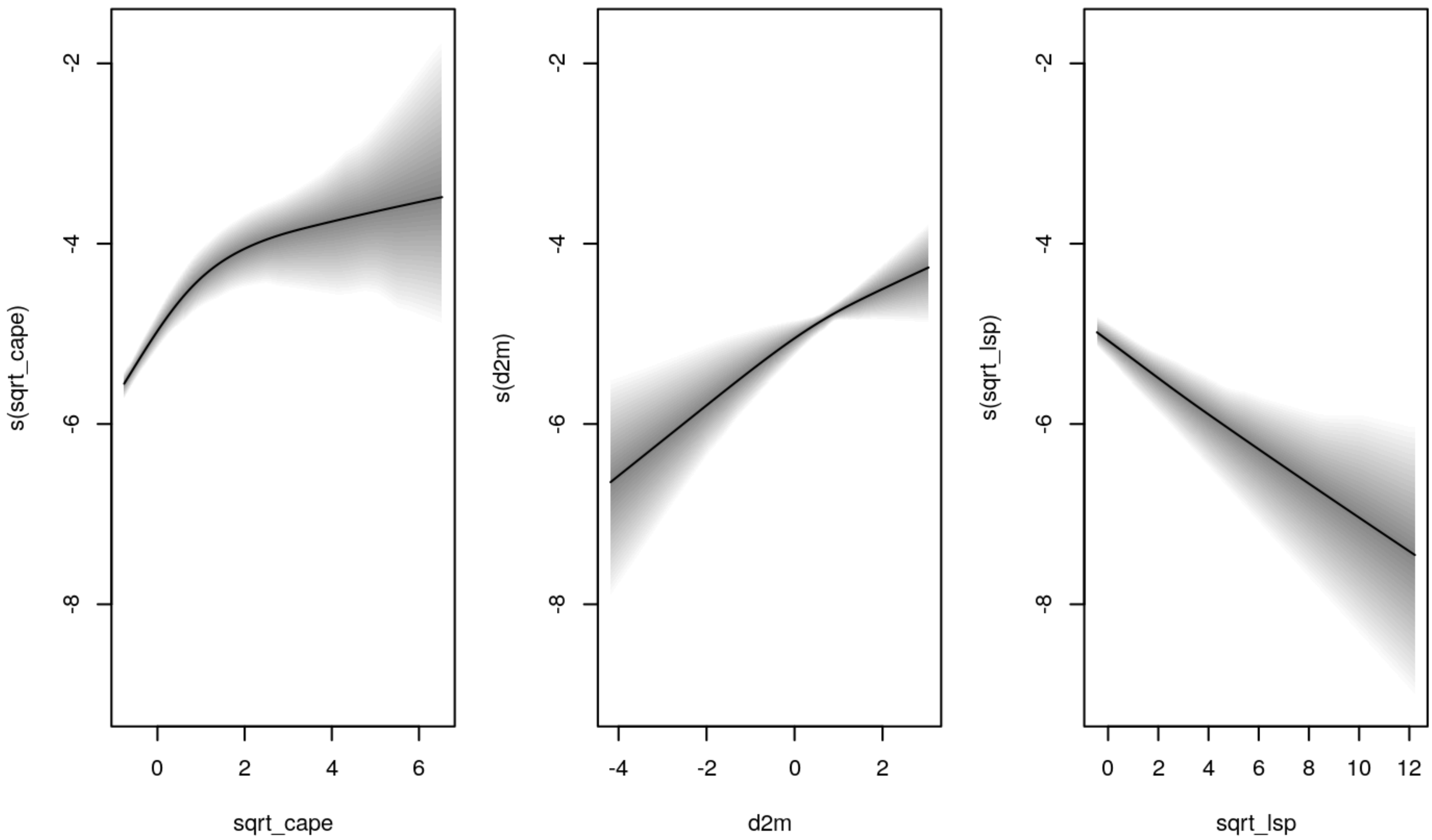
```
f <- list(
  counts ~ s(sqrt_cape) + s(d2m) + s(sqrt_lsp) + ... + n(fn),
  sigma ~ s(sqrt_cape) + s(d2m) + s(sqrt_lsp) + ... + n(fn)
)
```

Estimation: Batchwise boosting & bagging including NN.

```
b <- bamlss(f, family = "dgp", data = flash_train,
  optimizer = bbfit, nu = 0.05, batch_ids = c(5000, 4000),
  aic = TRUE, select = TRUE, ...)
```

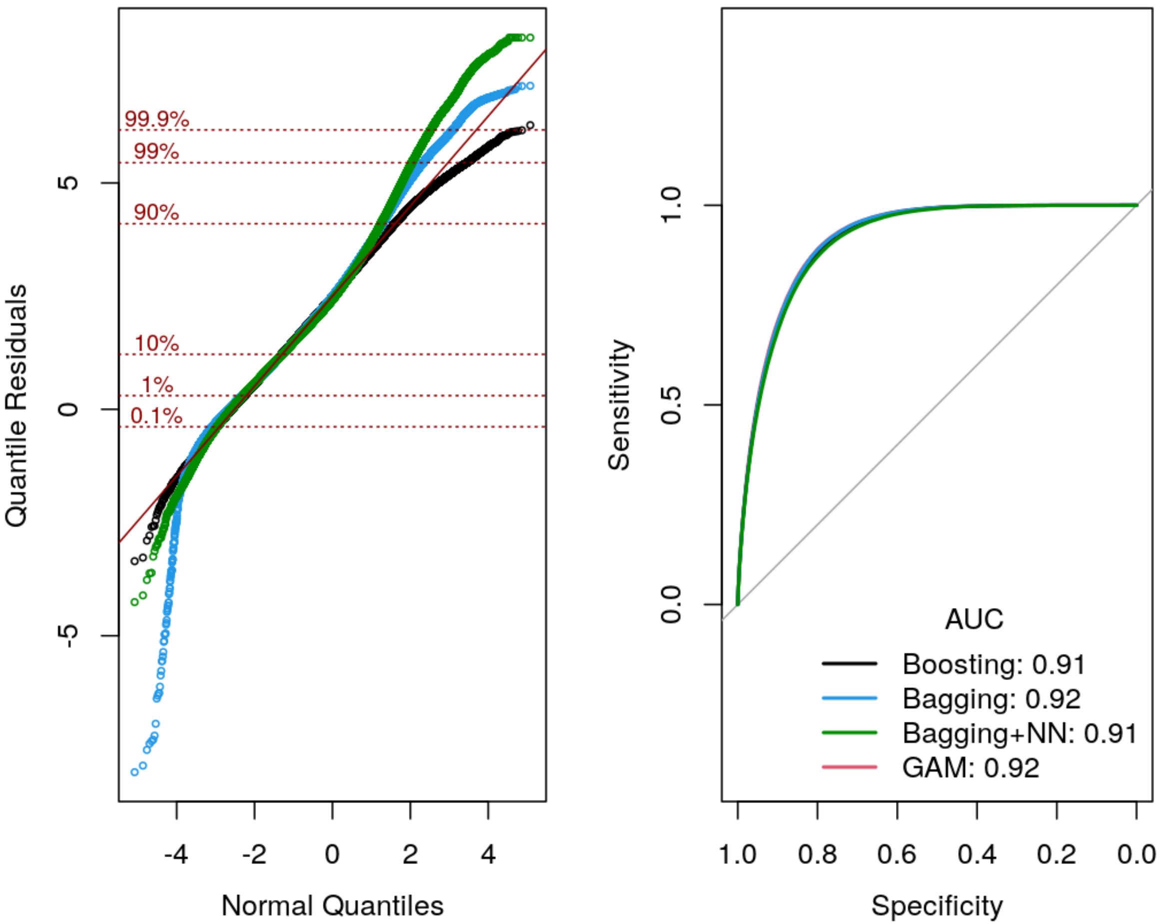
Lightning Model

```
plot(b, model = "xi", term = c("s(t2m)", "s(ssr)", "s(w_prof_PC2)"))
```



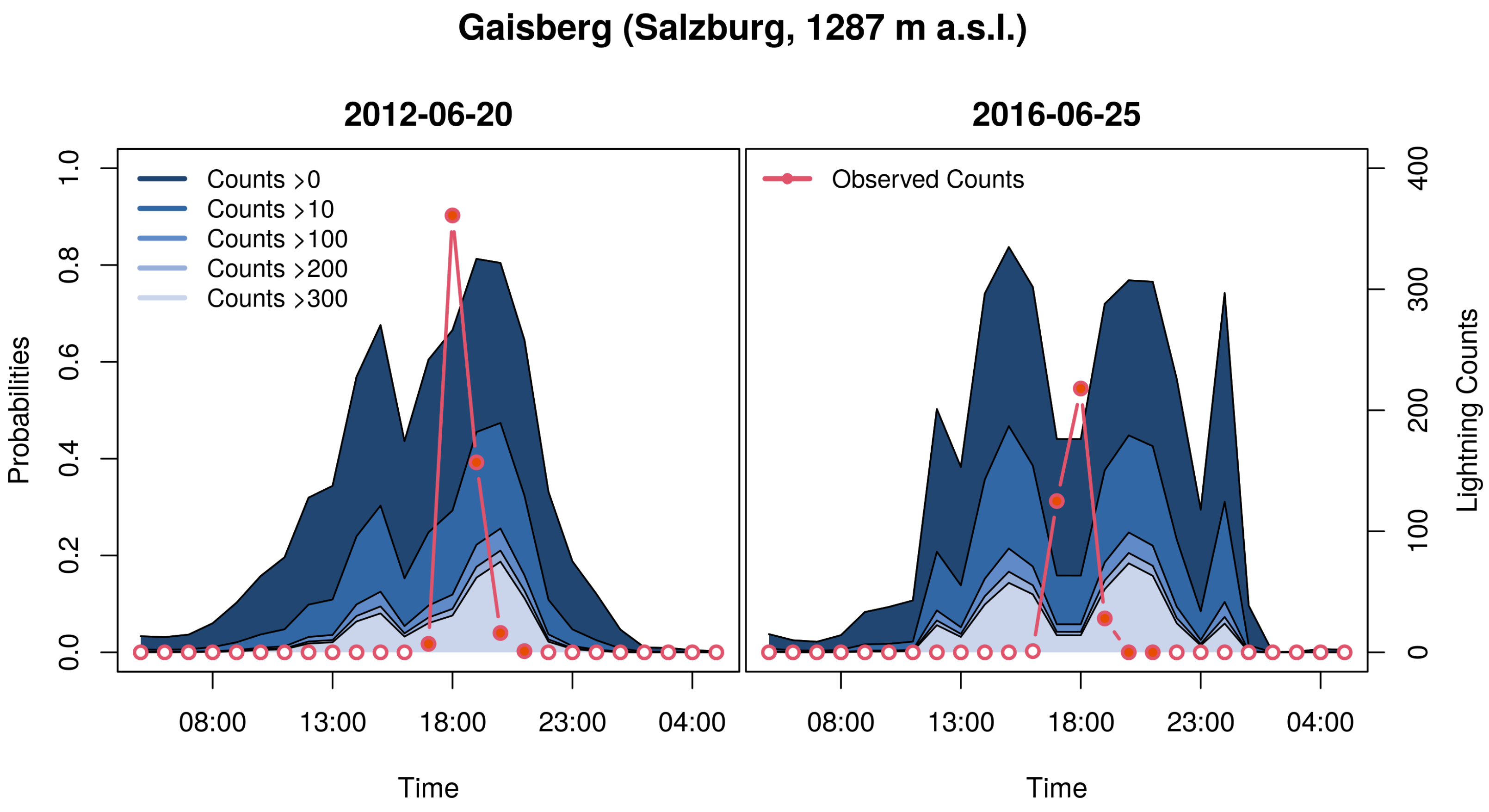
Lightning Model

```r
plot(b, which = "qq-resid")
```



Lightning Model

```r
p <- predict(b, newdata = nd, type = "parameter")
```



Lightning Model