Monotonicity-Constrained Regression:
An Application to Species Distribution Models

Benjamin Hofner

Institut für Medizininformatik, Biometrie und Epidemiologie (IMBE)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Biometrische Kolloquium 2012, Berlin

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
MEDIZINISCHE FAKULTÄT

Red Kite Breeding in Bavaria

- **Aim:** Model the probability of breeding for Red Kites *(Milvus milvus)* in Bavaria in order to understand the impact of environmental variables and climatic changes on species distribution
- **Response:** Red Kite breeding observed (yes/no)
- **Observations:** 2 periods each with 1918 observational cells of size 40 km^2 (3836 observations in total)
- **Observation periods:** 1979–1983 & 1996–1999
- **Predictors:** Characterizing climatic conditions and land cover

- Spatio-temporal logistic regression model

- **Possible predictors:**
  - 15 binary covariates (presence and absence of characteristics)
  - 1 ordinal covariate (fraction of cities and villages [0%, 1-10%, >10%])
  - 36 continuous variables

  and
  - Spatial information (longitude and latitude)
  - Temporal information (period of observation)
- Variable selection and model choice required
- **Additional challenge:**
  Some effects should be monotonic for biological reasons.

We will show how boosting ...
is able to fulfill all these requirements.

Boosting (in a Nutshell)

Model Fitting with Component-Wise Boosting

Structured Additive Model

$$\mu_i = \mathbb{E}(y|\boldsymbol{x}_i) = h(\eta_i(\boldsymbol{x}_i))$$

with response function h and **additive** predictor

$$\eta_i(\boldsymbol{x}_i) = \beta_0 + \sum_{j=1}^{J} f_j(\boldsymbol{x}_i),$$

- Model fitting aims at **minimizing the expected loss** with appropriate **loss function** ρ, e.g.,
  - for Gaussian model: **squared error loss** $\rho(y, \eta(\boldsymbol{x})) = (y - \eta(\boldsymbol{x}))^2$
  - for GLMs: **negative log-likelihood**
- In practice: Minimization of the **empirical risk**

$$n^{-1} \sum_{i=1}^{n} \rho(y_i, \eta_i(\boldsymbol{x}_i))$$

Boosting

- minimizes empirical risk (e.g., **negative log likelihood**)
- in a stagewise fashion
- via functional gradient descent (FGD).

In each iteration m

- the negative gradient of the loss function

$$u_i^{[m]} = - \left. \frac{\partial \rho(y_i, \eta)}{\partial \eta} \right|_{\eta = \hat{\eta}_i^{[m-1]}}$$

  is estimated via penalized least squares base-learners ($\hat{\boldsymbol{u}}^{[m]} = \hat{g}_j(\boldsymbol{x})$)
- update only model term corresponding to the **best-fitting base-learner** $\hat{g}_{j*}$ and add a **small fraction** ν **of the estimate** $\hat{g}_{j*}$ (e.g., 10%) to the model
- variable and model selection is achieved

Practical notes

- Base-learners represent functions $f_j(\cdot)$ from structured additive predictor.
- We get an interpretable model, similar to models from maximum likelihood estimation.
- Additionally, regularization is achieved via base-learner selection and shrinkage.

Practical notes

- Base-learners represent functions $f_j(\cdot)$ from structured additive predictor.
- We get an interpretable model, similar to models from maximum likelihood estimation.
- Additionally, regularization is achieved via base-learner selection and shrinkage.

Implementation

- All results discussed here are implemented in the R package **mboost** [Hothorn, Bühlmann, Kneib, Schmid, and Hofner 2010; 2012]
- The estimation problem is split into two (main) parts:
  - the loss function, which specifies the estimation problem ("`family`")
  - the base-learners, which specify the types of effects.

  Everything can be freely combined.

Constrained Regression

based on Hofner et al. [2011]

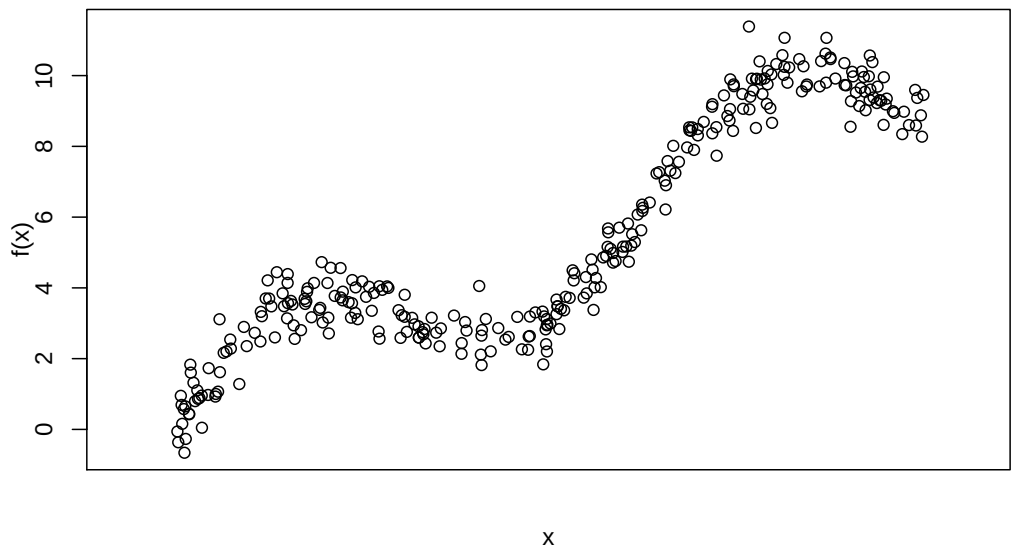
Short Review of P-splines

[Eilers and Marx 1996]

A **smooth function** can be expressed with B-splines as

$$f(x) = \sum_{j=1}^{J} \beta_j B_j(x; l) = \boldsymbol{\beta}^\top \boldsymbol{B}(x), \tag{1}$$

where $B_j(\cdot; l)$ is the j-th B-spline basis function of **degree** l defined on a **grid of knots** ξ_k.



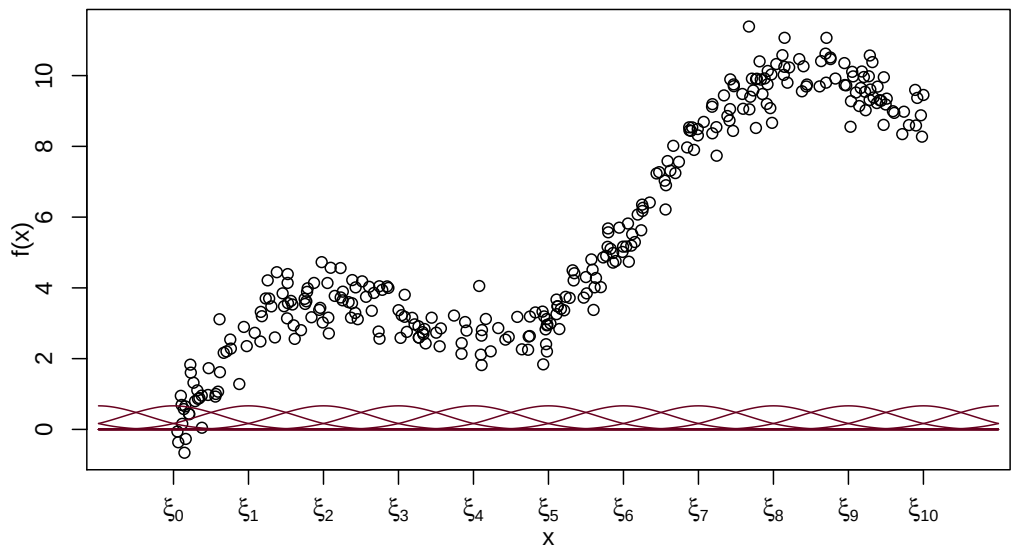
Short Review of P-splines

[Eilers and Marx 1996]

A **smooth function** can be expressed with B-splines as

$$f(x) = \sum_{j=1}^{J} \beta_j B_j(x; l) = \boldsymbol{\beta}^\top \boldsymbol{B}(x), \tag{1}$$

where $B_j(\cdot; l)$ is the j-th B-spline basis function of **degree** l defined on a **grid of knots** ξ_k.



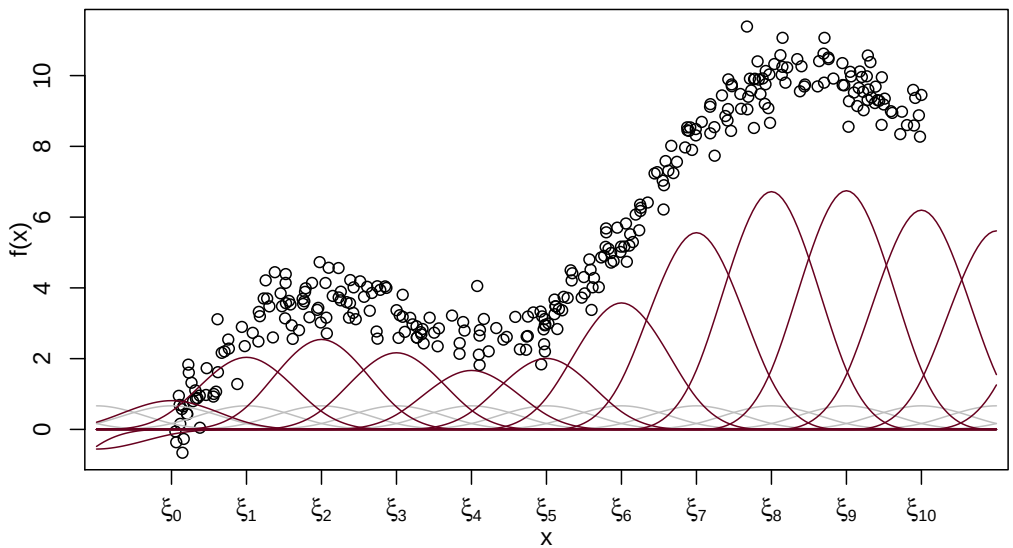
Short Review of P-splines

[Eilers and Marx 1996]

A **smooth function** can be expressed with B-splines as

$$f(x) = \sum_{j=1}^{J} \beta_j B_j(x; l) = \boldsymbol{\beta}^\top \boldsymbol{B}(x), \tag{1}$$

where $B_j(\cdot; l)$ is the j-th B-spline basis function of **degree** l defined on a **grid of knots** ξ_k.



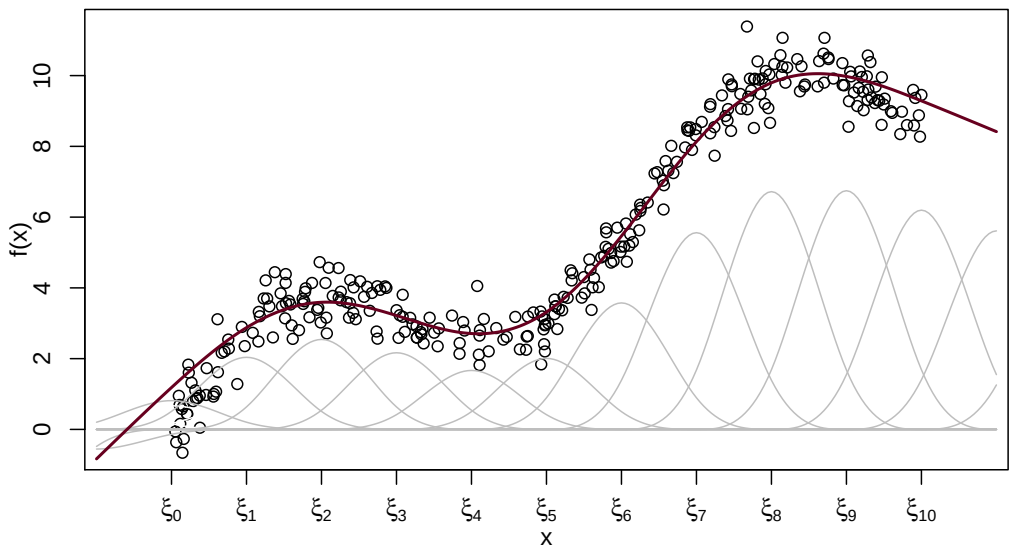
Short Review of P-splines

[Eilers and Marx 1996]

A **smooth function** can be expressed with B-splines as

$$f(x) = \sum_{j=1}^{J} \beta_j B_j(x; l) = \boldsymbol{\beta}^\top \boldsymbol{B}(x), \tag{1}$$

where $B_j(\cdot; l)$ is the j-th B-spline basis function of **degree** l defined on a **grid of knots** ξ_k.



- Smoothness enforced by additional **penalty on adjacent B-splines**:

$$\mathcal{J}(\boldsymbol{\beta}; d) = \sum_{j=d+1}^{J} (\Delta^d \beta_j)^2,$$

  where d is the order of the difference penalty.
- **Difference operator** Δ^d is defined as:

$$\Delta \beta_j = \Delta^1 \beta_j = (\beta_j - \beta_{j-1})$$

$$\Delta^2 \beta_j = \Delta(\Delta \beta_j) = \beta_j - 2\beta_{j-1} + \beta_{j-2}$$

- Smoothness enforced by additional **penalty on adjacent B-splines**:

$$\mathcal{J}(\boldsymbol{\beta}; d) = \sum_{j=d+1}^{J} (\Delta^d \beta_j)^2 \ = \boldsymbol{\beta}^\top \boldsymbol{D}_{(d)}^\top \boldsymbol{D}_{(d)} \boldsymbol{\beta},$$

  where d is the order of the difference penalty.
- **Difference operator** Δ^d is defined as:

$$\Delta \beta_j = \Delta^1 \beta_j = (\beta_j - \beta_{j-1})$$

$$\Delta^2 \beta_j = \Delta(\Delta \beta_j) = \beta_j - 2\beta_{j-1} + \beta_{j-2}$$

- **Difference matrix** $\boldsymbol{D}_{(d)}$

$$\boldsymbol{D}_{(1)} = \begin{pmatrix} -1 & 1 & 0 & \dots \\ 0 & -1 & 1 & \dots \\ 0 & 0 & \ddots & \ddots \end{pmatrix} \quad \boldsymbol{D}_{(2)} = \begin{pmatrix} 1 & -2 & 1 & 0 & \dots \\ 0 & 1 & -2 & 1 & \dots \\ 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix}$$

- Smoothness enforced by additional **penalty on adjacent B-splines**:

$$\mathcal{J}(\boldsymbol{\beta}; d) = \sum_{j=d+1}^{J} (\Delta^d \beta_j)^2 \ = \boldsymbol{\beta}^\top \boldsymbol{D}_{(d)}^\top \boldsymbol{D}_{(d)} \boldsymbol{\beta},$$

  where d is the order of the difference penalty.

- **Difference operator** Δ^d is defined as:

$$\Delta \beta_j = \Delta^1 \beta_j = (\beta_j - \beta_{j-1})$$

$$\Delta^2 \beta_j = \Delta(\Delta \beta_j) = \beta_j - 2\beta_{j-1} + \beta_{j-2}$$

- **Difference matrix** $\boldsymbol{D}_{(d)}$

$$\boldsymbol{D}_{(1)} = \begin{pmatrix} -1 & 1 & 0 & \dots \\ 0 & -1 & 1 & \dots \\ 0 & 0 & \ddots & \ddots \end{pmatrix} \quad \boldsymbol{D}_{(2)} = \begin{pmatrix} 1 & -2 & 1 & 0 & \dots \\ 0 & 1 & -2 & 1 & \dots \\ 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix}$$

Estimation: Penalized least squares criterion

$$\mathcal{Q}(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta})^\top (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta}) + \lambda \mathcal{J}(\boldsymbol{\beta}; d)$$

Monotonic P-splines

- ✓ **Smoothness Constraint:**
  Difference penalty on coefficients of adjacent B-splines
- **Monotonicity Constraint:**

$$f'(x) = \frac{\partial}{\partial x} \sum \beta_j B_j(x; l) = \frac{1}{h} \sum \Delta^1 \beta_{j+1} B_j(x; l-1)$$

(distance of knots $h > 0$, B-spline basis $B_j(x; l-1) \geq 0$)

  - depends only on the **first differences of the adjacent coefficients**

Monotonic P-splines

- ✓ **Smoothness Constraint:**
  Difference penalty on coefficients of adjacent B-splines
- **Monotonicity Constraint:**

$$f'(x) = \frac{\partial}{\partial x} \sum \beta_j B_j(x; l) = \frac{1}{h} \sum \Delta^1 \beta_{j+1} B_j(x; l-1)$$

(distance of knots $h > 0$, B-spline basis $B_j(x; l-1) \geq 0$)

  - depends only on the **first differences of the adjacent coefficients**
  - Monotonic increasing function:

$$\Delta^1 \beta_{j+1} > 0 \quad \forall j \quad \Rightarrow \quad f'(x) > 0$$

Monotonic P-splines

- ✓ **Smoothness Constraint:**
  Difference penalty on coefficients of adjacent B-splines
- **Monotonicity Constraint:**

$$f'(x) = \frac{\partial}{\partial x} \sum \beta_j B_j(x; l) = \frac{1}{h} \sum \Delta^1 \beta_{j+1} B_j(x; l-1)$$

(distance of knots $h > 0$, B-spline basis $B_j(x; l-1) \geq 0$)

  - depends only on the **first differences of the adjacent coefficients**
  - Monotonic increasing function:

$$\Delta^1 \beta_{j+1} > 0 \quad \forall j \quad \Rightarrow \quad f'(x) > 0$$

  - Monotonic decreasing function:

$$\Delta^1 \beta_{j+1} < 0 \quad \forall j \quad \Rightarrow \quad f'(x) < 0$$

Monotonic P-splines

- ✓ **Smoothness Constraint:**
  Difference penalty on coefficients of adjacent B-splines
- **Monotonicity Constraint:**

$$f'(x) = \frac{\partial}{\partial x} \sum \beta_j B_j(x; l) = \frac{1}{h} \sum \Delta^1 \beta_{j+1} B_j(x; l-1)$$

(distance of knots $h > 0$, B-spline basis $B_j(x; l-1) \geq 0$)

  - depends only on the **first differences of the adjacent coefficients**
  - Monotonic increasing function:

$$\Delta^1 \beta_{j+1} > 0 \quad \forall j \quad \Rightarrow \quad f'(x) > 0$$

  - Monotonic decreasing function:

$$\Delta^1 \beta_{j+1} < 0 \quad \forall j \quad \Rightarrow \quad f'(x) < 0$$

- **Convexity / Concavity Constraint:**

$$f''(x) = \frac{1}{h^2} \sum \Delta^2 \beta_{j+2} B_j(x; l-2)$$

Penalty

only needed for differences that violate the monotonicity assumption

- "Monotonic coefficients" can be achieved by (additional) **asymmetric difference penalties** on adjacent coefficients [Eilers 2005]:

$$\mathcal{J}_{\text{asym}}(\boldsymbol{\beta}; c) = \sum_{j=c+1}^{J} v_j (\Delta^c \beta_j)^2, = \boldsymbol{\beta}^\top \boldsymbol{D}_{(c)}^\top \boldsymbol{V} \boldsymbol{D}_{(c)} \boldsymbol{\beta}, \tag{2}$$

where c is the order of the difference penalty.

- "Monotonic coefficients" can be achieved by (additional) **asymmetric difference penalties** on adjacent coefficients [Eilers 2005]:

$$\mathcal{J}_{\text{asym}}(\boldsymbol{\beta}; c) = \sum_{j=c+1}^{J} v_j (\Delta^c \beta_j)^2, = \boldsymbol{\beta}^\top \boldsymbol{D}_{(c)}^\top \boldsymbol{V} \boldsymbol{D}_{(c)} \boldsymbol{\beta}, \tag{2}$$

  where c is the order of the difference penalty.

- Important difference to P-spline penalty are **weights** v_j, which are specified as

$$v_j = \begin{cases} 0 & \text{if } \Delta^c \beta_j > 0 \\ 1 & \text{if } \Delta^c \beta_j \leq 0, \end{cases} \quad \text{monotonic increasing} \tag{3}$$

  matrix notation: $\boldsymbol{V} = \text{diag}(\boldsymbol{v})$, and difference matrices as above.

- "Monotonic coefficients" can be achieved by (additional) **asymmetric difference penalties** on adjacent coefficients [Eilers 2005]:

$$\mathcal{J}_{\text{asym}}(\boldsymbol{\beta}; c) = \sum_{j=c+1}^{J} v_j (\Delta^c \beta_j)^2, = \boldsymbol{\beta}^\top \boldsymbol{D}_{(c)}^\top \boldsymbol{V} \boldsymbol{D}_{(c)} \boldsymbol{\beta}, \tag{2}$$

  where c is the order of the difference penalty.

- Important difference to P-spline penalty are **weights** v_j, which are specified as

$$v_j = \begin{cases} 0 & \text{if } \Delta^c \beta_j < 0 \\ 1 & \text{if } \Delta^c \beta_j \geq 0, \end{cases} \text{ monotonic decreasing} \tag{3}$$

  matrix notation: $\boldsymbol{V} = \text{diag}(\boldsymbol{v})$, and difference matrices as above.

- "Monotonic coefficients" can be achieved by (additional) **asymmetric difference penalties** on adjacent coefficients [Eilers 2005]:

$$\mathcal{J}_{\text{asym}}(\boldsymbol{\beta}; c) = \sum_{j=c+1}^{J} v_j (\Delta^c \beta_j)^2, = \boldsymbol{\beta}^\top \boldsymbol{D}_{(c)}^\top \boldsymbol{V} \boldsymbol{D}_{(c)} \boldsymbol{\beta}, \tag{2}$$

  where c **is the order of the difference penalty**.

- Important difference to P-spline penalty are weights v_j, which are specified as

$$v_j = \begin{cases} 0 & \text{if } \Delta^c \beta_j < 0 \\ 1 & \text{if } \Delta^c \beta_j \geq 0, \end{cases} \text{ monotonic decreasing} \tag{3}$$

  matrix notation: $\boldsymbol{V} = \text{diag}(\boldsymbol{v})$, and difference matrices as above.

- Monotonicity constraint if $c = 1$
- Convex / concave constraint if $c = 2$

Estimation of Constrained P-splines

Weights v_j depend on $\boldsymbol{\beta}$. Thus:

1) Start with standard P-spline estimate
2) Compute weights v_j for $\hat{\boldsymbol{\beta}}$
3) Minimize

$$\mathcal{Q}(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta})^\top (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta}) + \lambda_1 \mathcal{J}(\boldsymbol{\beta}; d) + \lambda_2 \mathcal{J}_{\text{asym}}(\boldsymbol{\beta}; c) \quad (4)$$

- **penalized least squares** estimate

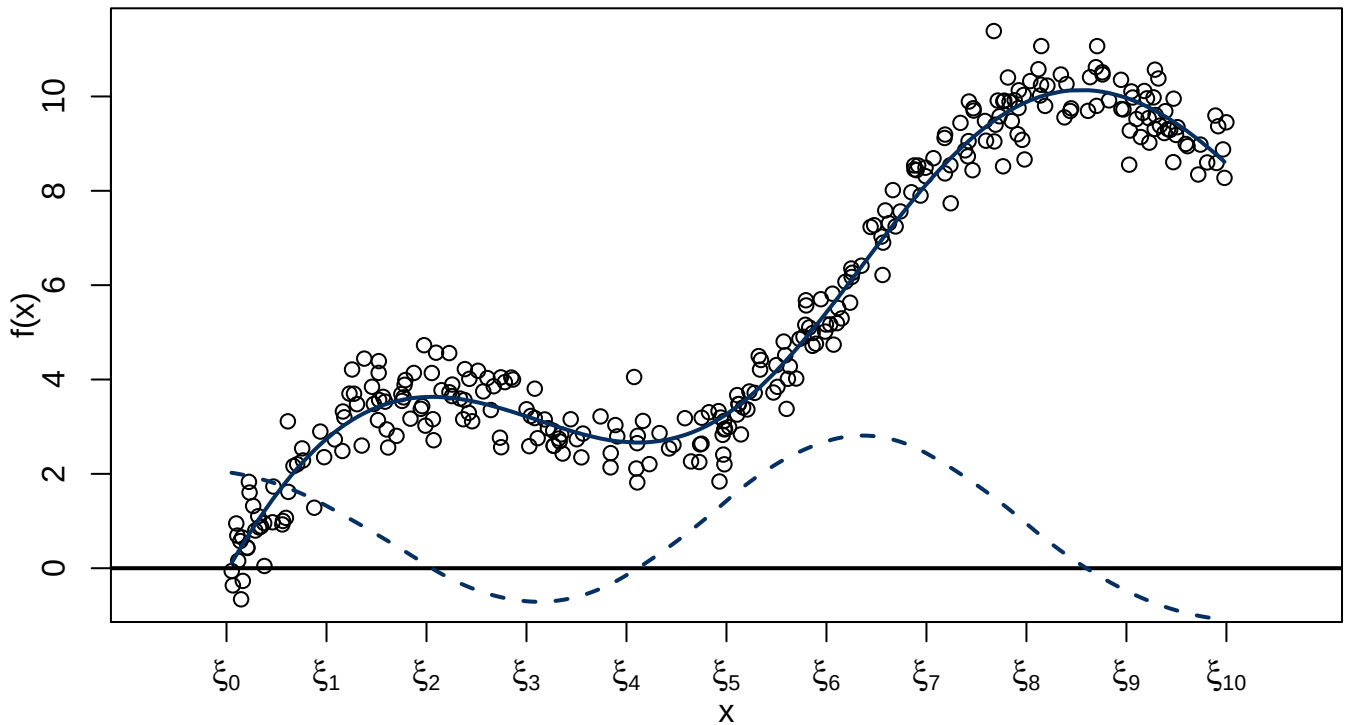
$$\hat{\boldsymbol{\beta}} = (\boldsymbol{B}^\top \boldsymbol{B} + \lambda_1 \boldsymbol{D}_{(d)}^\top \boldsymbol{D}_{(d)} + \lambda_2 \boldsymbol{D}_{(c)}^\top \boldsymbol{V} \boldsymbol{D}_{(c)})^{-1} \boldsymbol{B}^\top \boldsymbol{y}$$

4) Recompute weights v_j with $\hat{\boldsymbol{\beta}}$
5) Iterate 3) and 4) until no more changes in v_j (usually after 2-3 steps)

Smoothing parameter λ_2 is chosen quite large (e.g., 10^6), where larger values are associated with a stronger impact of the monotonic constraint

P-Splines, Monotonic Splines

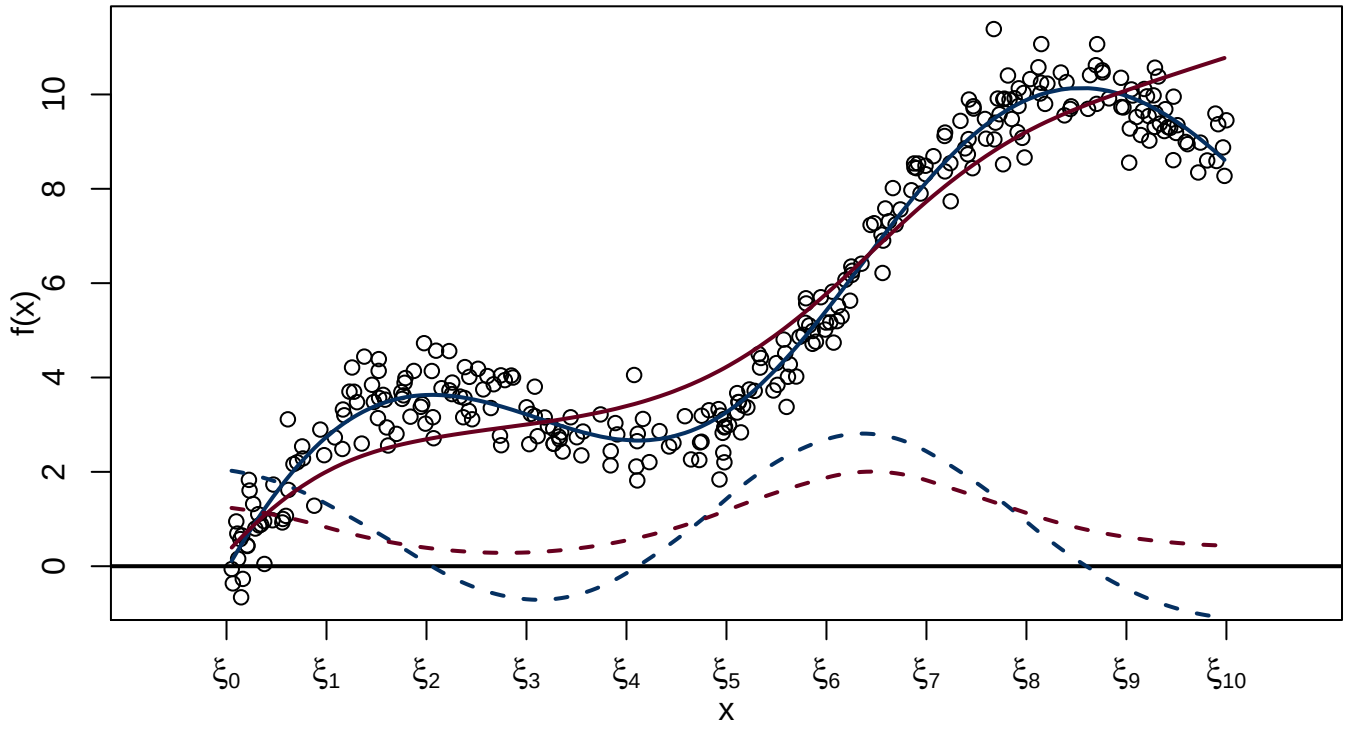
and their Derivatives



- P-spline estimate and derivative (dashed)

P-Splines, Monotonic Splines

and their Derivatives



- P-spline estimate and derivative (dashed)
- Monotonic spline estimate and derivative (dashed)

Incorporating Monotonic Effects into Boosting

- We use one base-learner per monotonic effect.
- The constrained estimation is done completely within the base-learner.
- There is no need to change the boosting algorithm itself.
- Monotonic effects can be freely combined with other (complex) types of effects in one model.

Incorporating Monotonic Effects into Boosting

- We use one base-learner per monotonic effect.
- The constrained estimation is done completely within the base-learner.
- There is no need to change the boosting algorithm itself.
- Monotonic effects can be freely combined with other (complex) types of effects in one model.

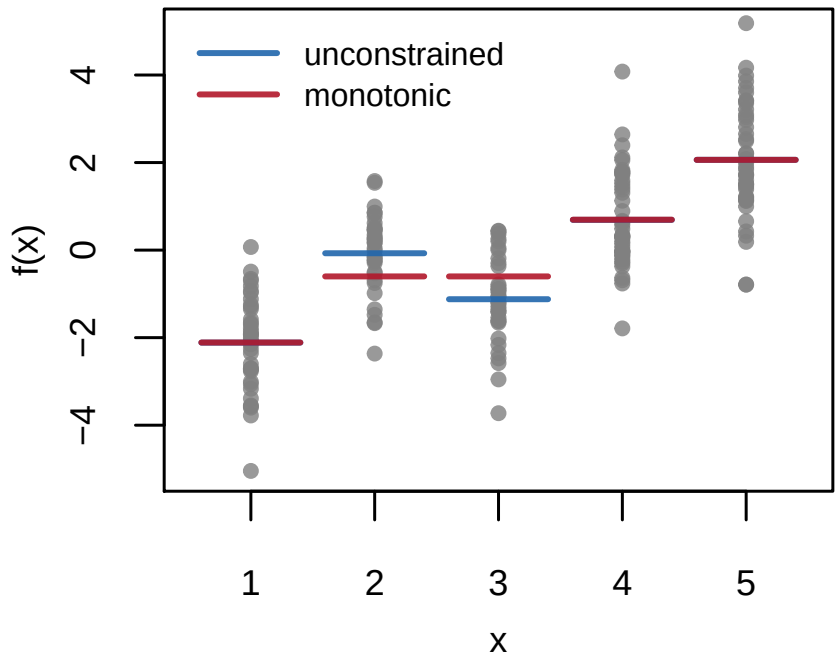
Simulation Results:

- We simulated models with two monotonic effects.
- We estimated a monotonicity constrained model (**mGAM**) and an unconstrained model (**GAM**).
- **mGAM** had better prediction accuracy
- **mGAM** did not violate monotonicity
- **GAM** most often violated monotonicity (even with a certain threshold)

For details see Hofner et al. [2011].

Further Constraints

- Monotonicity constraints can also be imposed on **ordinal, categorical variables** using the same idea.



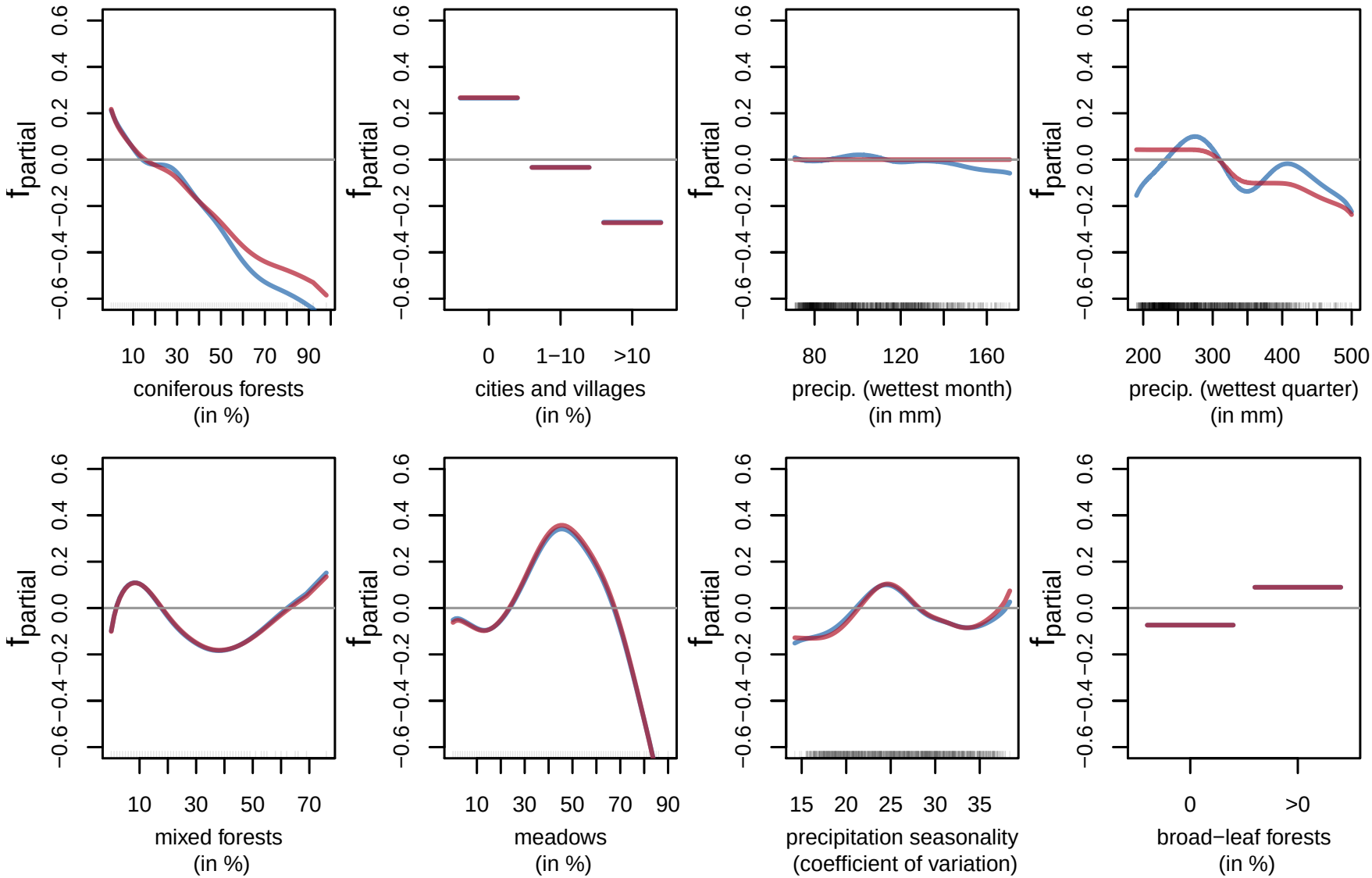
Application: Red Kite Breeding Distribution

Let us come back to have a look at the Red Kite breeding data.

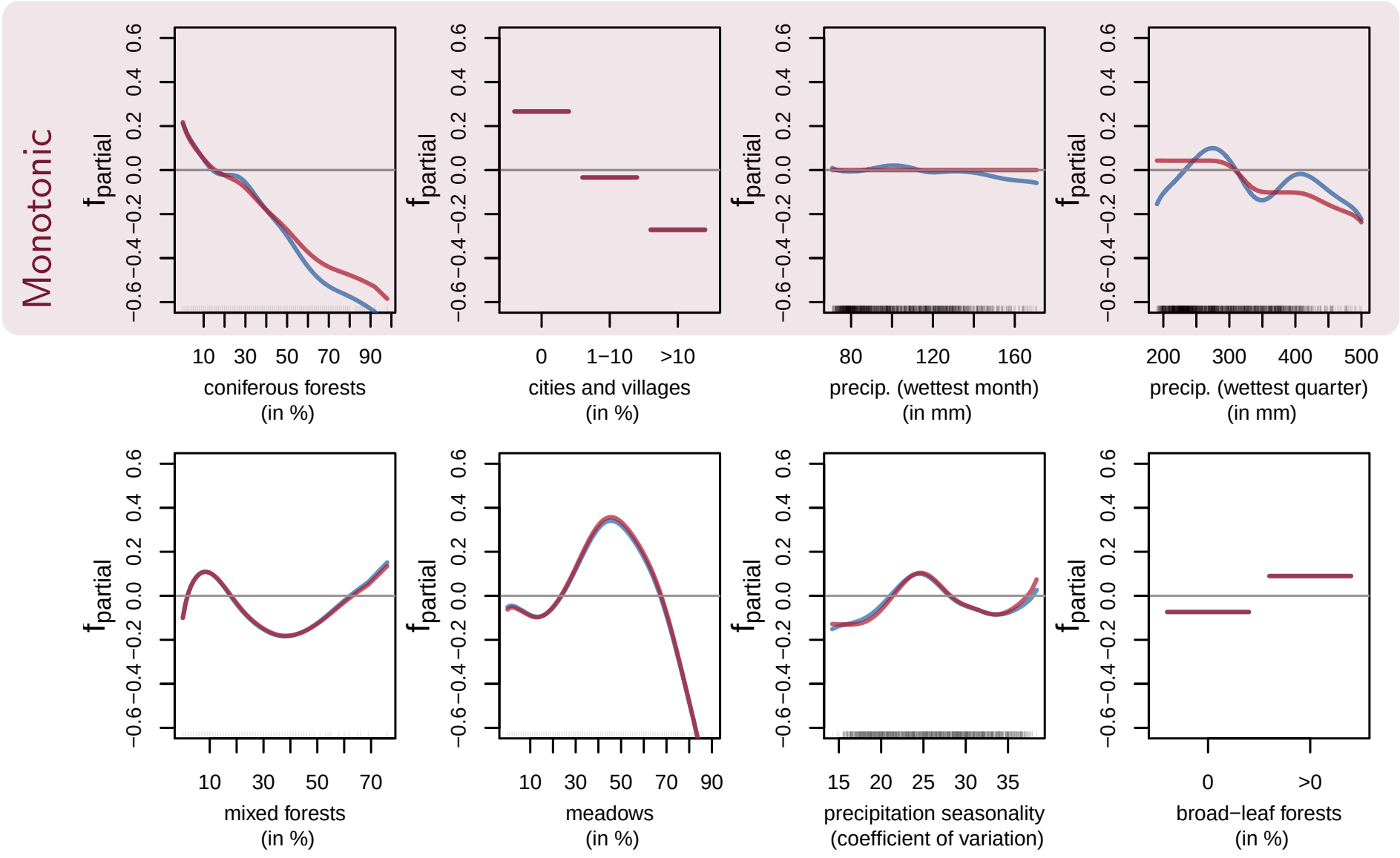
Aims:

- Model the breeding probability of Red Kites in Bavaria
- Following subject-matter knowledge (▶ Jörg Müller) we assume monotonically decreasing effects for
  - coniferous forests (continuous; in %)
  - cities and villages (ordinal; in %)
  - precipitation wettest month (continuous; in mm)
  - precipitation wettest quarter (continuous; in mm)
- Include variable selection (as we have 55 base-learners in total)

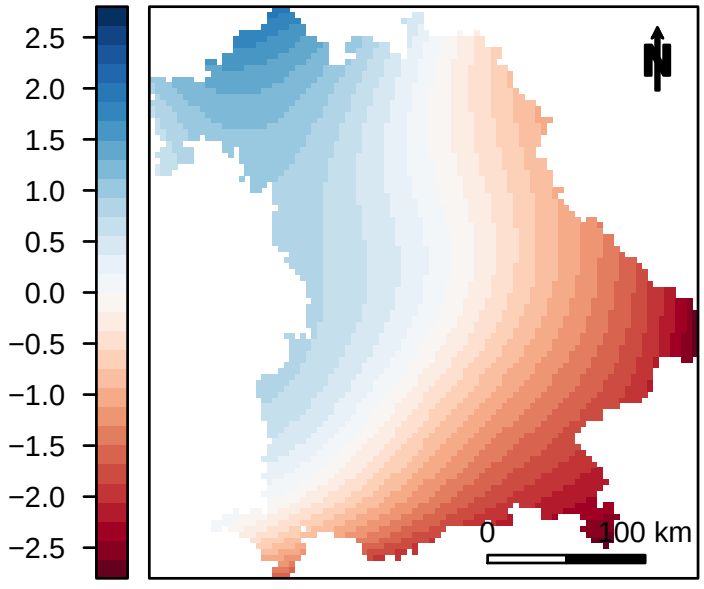
Results (selection)



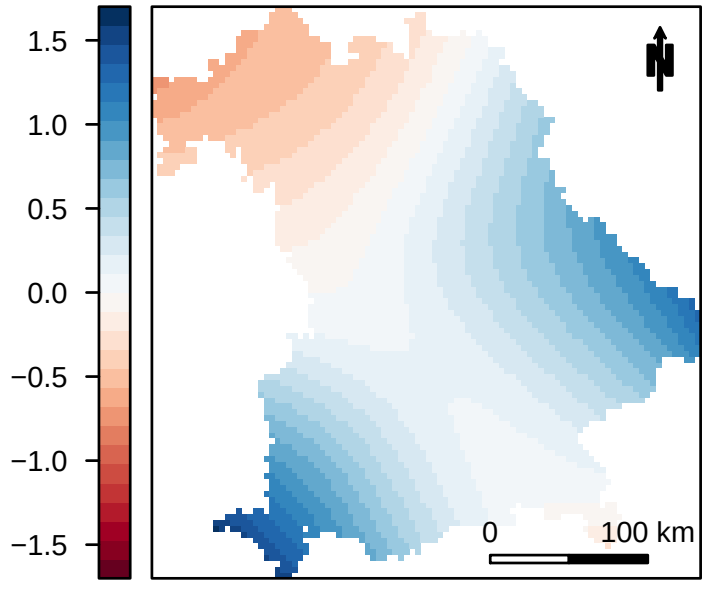
Results (selection)



Results (selection)



Spatial Effect (1979–1983)



Change of Spatial Effect
(between 1979–1983 and 1996–1999)

Model Summary

- Sparse model (only 10 out of 55 base-learners selected)
- Monotonicity constraints lead to models that are better to interpret
- Complex effects can be estimated within the same model (e.g., spatial or spatio-temporal effects and other non-restricted smooth effects)
- Other effects hardly affected (here)

Remark

- Constrained effects can be extended to **bivariate** monotonic effects

Summary

- Boosting allows to incorporate constrained effect estimates
  - No need to change the algorithm
  - Implementation using base-learners only
  - Can be combined with arbitrary loss functions and arbitrary other base-learners

R-package **mboost available on CRAN** to fit all the effects covered in this talk (and many more) in a very wide class of models [Hothorn et al. 2010; 2012]

- **New** tutorial available; see Hofner et al. [2012]
  and `vignette(package = "mboost", "mboost_tutorial")`

Outlook

- Partial monotonicity constraints (only on a certain subspace)
  Use only a subset of the knots in the asymmetric difference penalty
- Use constraints on boundaries (e.g., constant boundaries for effects that are supposed to level out)
  Use an additional penalty only for the boundaries with a pre-specified large penalty parameter
- Cyclic P-splines for time-series models
  "Wrap" basis functions and penalty matrix
- Adapt cyclic effects to ordinal variables (e.g., day of the week)
  Use a wrapped difference penalty

References

PHC Eilers. Unimodal smoothing. *Journal of Chemometrics*, 19:317–328, 2005.

PHC Eilers and BD Marx. Flexible smoothing with B-splines and penalties (with discussion). *Statistical Science*, 11:89–121, 1996.

B Hofner, J Müller, and T Hothorn. Monotonicity-constrained species distribution models. *Ecology*, 92:1895–1901, 2011.

B. Hofner, A. Mayr, N. Robinzonov, and M. Schmid. Model-based boosting in R – A hands-on tutorial using the R package mboost. Technical report, Department of Statistics, Ludwig-Maximilians-Universität München, 2012. URL http://epub.ub.uni-muenchen.de/12754/.

T Hothorn, P Bühlmann, T Kneib, M Schmid, and B Hofner. Model-based boosting 2.0. *Journal of Machine Learning Research*, 11:2109–2113, 2010.

T Hothorn, P Bühlmann, T Kneib, M Schmid, and B Hofner. *mboost: Model-Based Boosting*, 2012. URL http://CRAN.R-project.org/package=mboost. R package version 2.1-2.