Modern approaches for component-wise boosting:

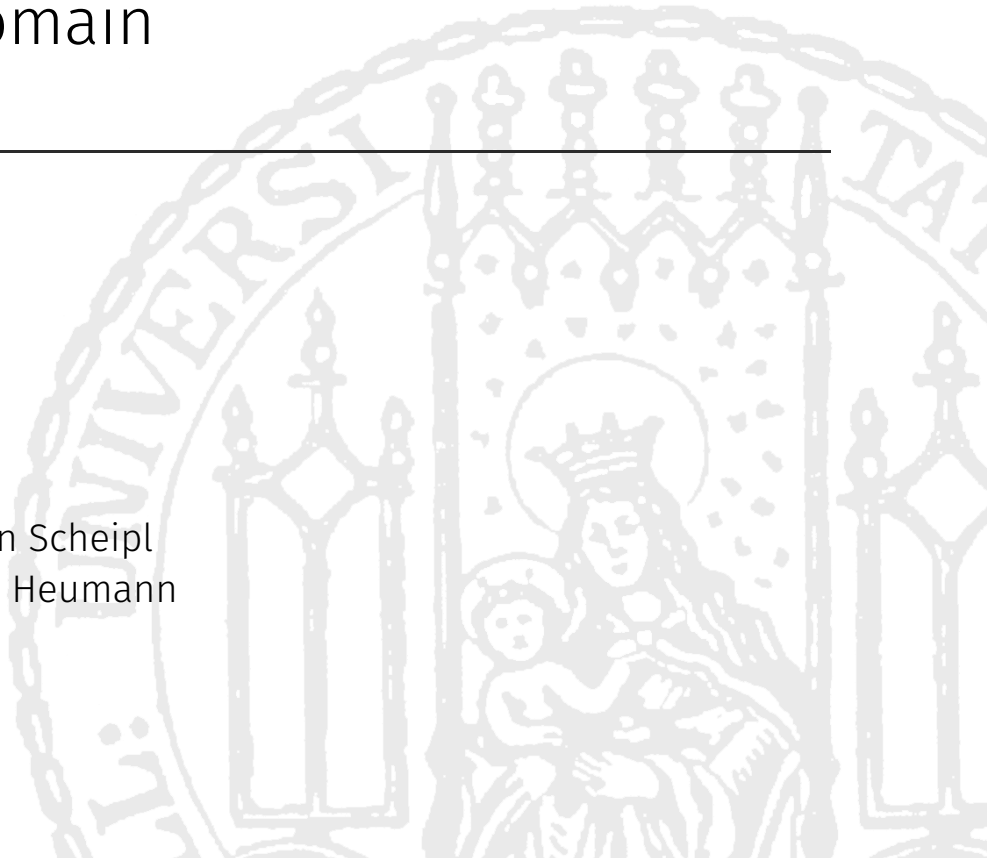
Automation, efficiency, and distributed computing with application to the medical domain

---

Daniel Schalk

March 24, 2023

Supervisor: Prof. Dr. Bernd Bischl
Reviewers: Prof. Dr. Matthias Schmid, PD Dr. Fabian Scheipl
Chair of the examination panel: Prof. Dr. Christian Heumann



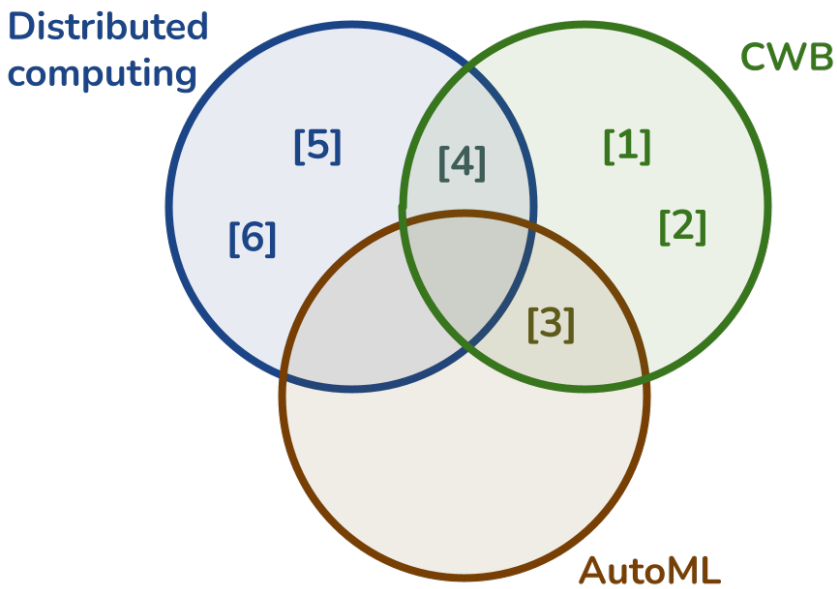
About the dissertation

Overview

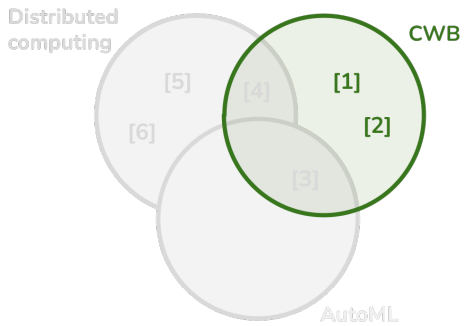
Focus of the dissertation:

Discuss and propose new and modern directions for component-wise gradient boosting (CWB; Bühlmann and Yu, 2003).

Contribution and topics:



Part I - Efficiency



Goal: Increase CWB's efficiency

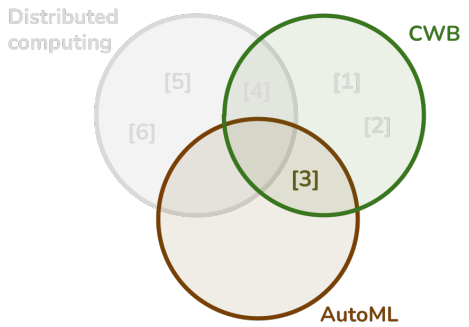
- **Acceleration:** Speed up the fitting process by using Nesterovs momentum.
- **Memory:** Reduce the memory consumption by discretizing numerical features.

Publications:

[1] Schalk, D., Thomas, J., and Bischl, B. (2018). compboost: Modular framework for component-wise boosting. *Journal of Open Source Software*, 3(30):967

[2] Schalk, D., Bischl, B., and Rügamer, D. (2022a). Accelerated componentwise gradient boosting using efficient data representation and momentum-based optimization. *Journal of Computational and Graphical Statistics*

Part II - Interpretable AutoML Framework



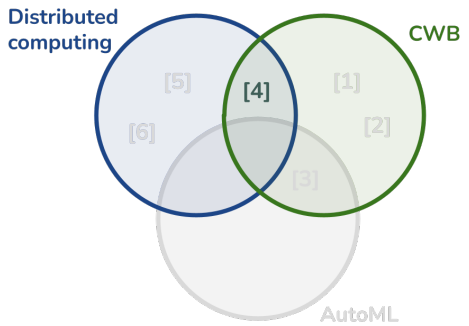
Goal:

- Easy access to an interpretable AutoML framework based on CWB as fitting engine.
- Focus is the assessment of the required complexity to model a given task.

Publication:

[3] Coors, S., Schalk, D., Bischl, B., and Rügamer, D. (2021). Automatic componentwise boosting: An interpretable automl system. *ECML-PKDD Workshop on Automating Data Science*

Part III - Distributed Computing



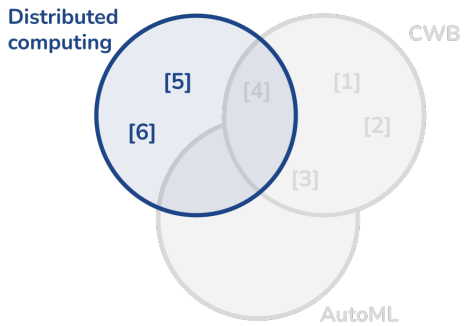
Goal:

- Distributed and privacy-preserving computation of CWB.
- Estimation of common shared effects and site-specific effect corrections.

Publications:

[4] Schalk, D., Bischl, B., and Rügamer, D. (2023a). Privacy-preserving and lossless distributed estimation of high-dimensional generalized additive mixed models. *arXiv preprint arXiv:2210.07723*. [Currently under review in the journal *Statistics and Computing*]

Part III - Distributed Computing



Goal:

- Methodology and implementation of a distributed and privacy-preserving ROC analysis.

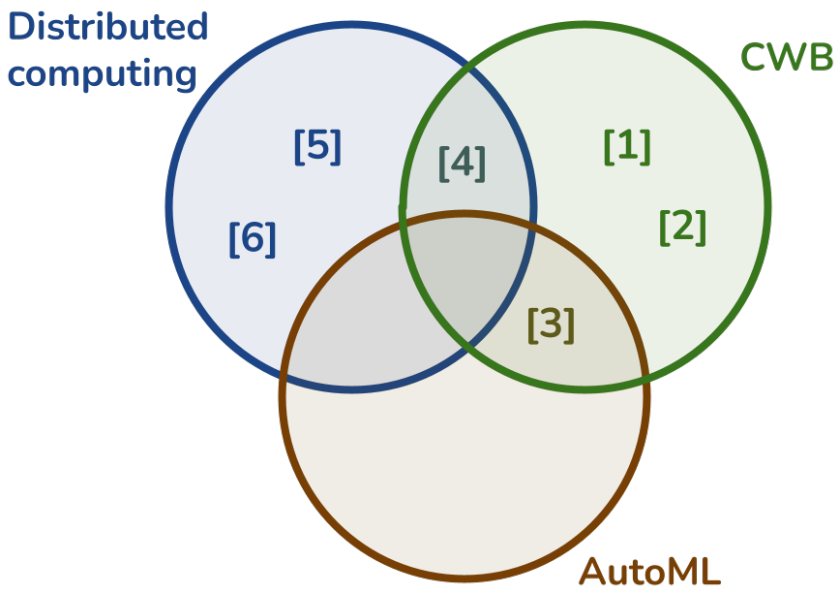
Publications:

[5] Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2022b). Distributed non-disclosive validation of predictive models by a modified roc-glm. *arXiv preprint arXiv:2203.10828* [Currently under review in the journal *BMC Medical Research Methodology*]

[6] Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2023b). dsBinVal: Conducting distributed roc analysis using datashield. *Journal of Open Source Software*, 8(82):4545

Overview

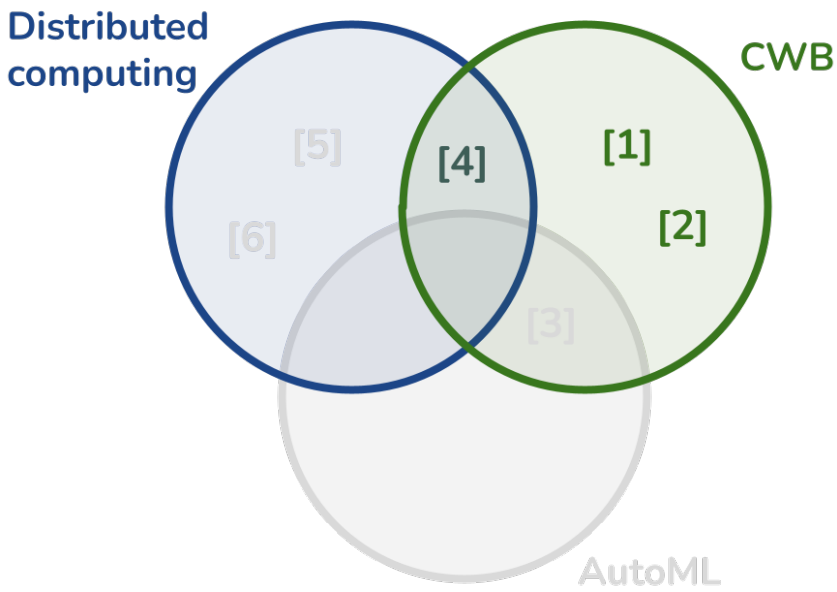
Contribution and topics:



The focus of this presentation is on CWB adjustments [1], [2], and [4]. The other contributions are briefly summarized at the end of the presentation.

Overview

Contribution and topics:



The focus of this presentation is on CWB adjustments [1], [2], and [4]. The other contributions are briefly summarized at the end of the presentation.

Structure of the talk

Background

Efficiency

Distributed computing

Automation

Distributed model evaluation

`compboost`

Background

History of component-wise boosting

Terminology

- p-dimensional covariate or feature vector $\mathbf{x} = (x_1, \ldots, x_p) \in \mathcal{X} = \mathcal{X}_1 \times \cdots \times$ and target variable $y \in \mathcal{Y}$.
- Data set $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}) \mid i = 1, \ldots, n\}$ with $(\mathbf{x}^{(i)}, y^{(i)})$ sampled from an unknown probability distribution $\mathbb{P}_{xy}$.
- True underlying relationship $f : \mathcal{X}^p \to \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$.
- Goal of Machine Learning (ML) is to estimate a model $\hat{f} = \arg\min_f \mathcal{R}_{\text{emp}}(f|\mathcal{D})$ with
  - Empirical risk $\mathcal{R}_{\text{emp}}(f|\mathcal{D}) = n^{-1} \sum_{(\mathbf{x},y)\in\mathcal{D}} L(y, \hat{f}(\mathbf{x}))$ and
  - Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, $(y, \hat{y}) \mapsto L(y, \hat{y})$.
- The inducer $\mathcal{I} : \mathbb{D} \times \Lambda \to \mathcal{F}$, $(\mathcal{D}, \boldsymbol{\lambda}) \mapsto \hat{f} = \mathcal{I}_{\boldsymbol{\lambda}}(\mathcal{D})$ gets a data set $\mathcal{D} \in \mathbb{D}$ with hyperparameters (HPs) $\boldsymbol{\lambda} \in \Lambda$.

Gradient boosting

- Gradient boosting (GB) aims to estimate f based on assembling weak base learners $b : \mathcal{X} \to \mathcal{Y}, \mathbf{x} \mapsto b(\mathbf{x}|\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$.
- The model estimate $\hat{f}$ is fitted by conducting functional gradient descent $\hat{f}^{[m-1]} = \hat{f}^{[m]} + \nu \hat{b}^{[m]}$ for M steps. The estimated model is then $\hat{f} = \hat{f}^{[M]}$.
- To obtain the model update $\hat{b}^{[m]}$ in iteration m, the weak base learner b is fit to pseudo residuals $\boldsymbol{r}^{[m]}$ by minimizing the SSE: $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
- The pseudo residuals $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f=\hat{f}^{[m-1]}}$, $i \in \{1, \ldots, n\}$, ($\boldsymbol{r}^{[m]}$ is the vector of pseudo residuals) contain the information in which direction to move $\hat{f}^{[m]}$ for a better fit to the training data $\mathcal{D}$.
- The fitting is initialized with $\hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$ and repeated M times or until an early stopping criterion is met.

Gradient boosting – Algorithm

Algorithm 1 GB algorithm

Input Train data $\mathcal{D}$, number of boosting iterations M, loss function L, base learner b
Output Model $\hat{f} = \hat{f}^{[M]}$

procedure GB($\mathcal{D}, M, L, b$)
  Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \textbf{arg min}_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$
  while $m \leq M$ **do**
    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$
    $\hat{\boldsymbol{\theta}}^{[m]} = \textbf{arg min}_{\boldsymbol{\theta}} \sum_{i=1}^{n} (r^{[m](i)} - b(\mathbf{x}^{(i)}|\boldsymbol{\theta}))^2$
    $\nu_m = \textbf{arg min}_{\nu \in \mathbb{R}} \sum_{i=1}^{n} L(r^{[m](i)}, \hat{f}^{[m]} + \nu \hat{b}^{[m]}(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}^{[m]}))$
    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu_m \hat{b}^{[m]}(\mathbf{x}|\hat{\boldsymbol{\theta}}^{[m]})$
  return $\hat{f} = \hat{f}^{[M]}$

Note: A common choice for the base learner in GB is, e.g., to use trees (Friedman, 2001). Based on the base learner, further adaptions to the algorithm are made to, e.g., increase speed or predictive power (Chen et al., 2015).

Component-wise gradient boosting – Basics

- Compared to GB, CWB can choose from a set of K base learners $b \in \{b_1, \ldots, b_K\}$.
- The learning rate ν is fixed and not optimized by a line search.
- Often, $b_1, \ldots, b_K$ are chosen to be (interpretable) statistical models and hence f corresponds to a generalized additive model (GAM; Hastie, 2017):

$$f(\mathbf{x}) = f_0 + \sum_{k=1}^{K} b_k(\mathbf{x}), \quad \text{intercept } f_0$$

- Advantages of CWB:
  - Feasible to get fit in high-dimensional feature spaces ($p \gg n$).
  - An inherent (unbiased) feature selection.
  - Interpretable/explainable partial feature effects (depending on the choice of base learners).

Component-wise gradient boosting – Base learner

- From now on, each base learner b_k is defined by a basis transformation $g_k : \mathcal{X} \to \mathbb{R}^{d_k}$ with $g_k(\mathbf{x}) = (g_{k,1}(\mathbf{x}), \ldots, g_{k,d_k}(\mathbf{x}))^\top$.
- The base learners are also restricted to be linear in the parameters: $b_k(\mathbf{x}|\boldsymbol{\theta}) = g_k(\mathbf{x})^\top \boldsymbol{\theta}$
- Due to the linearity, the sum of two base learners $b_k(\mathbf{x}|\boldsymbol{\theta}_l) + b_k(\mathbf{x}|\boldsymbol{\theta}_m)$ equals $b_k(\mathbf{x}|\boldsymbol{\theta}_l + \boldsymbol{\theta}_m)$.
- For n data points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, each base learner defines a design matrix $Z_k = (g_k(\mathbf{x}^{(1)})^\top, \ldots, g_k(\mathbf{x}^{(n)})^\top)^\top \in \mathbb{R}^{n \times d_k}$.
- Based on the linearity and the design matrix, each base learner can be fitted by calculating the least squares estimator $\hat{\boldsymbol{\theta}}_k = (Z_k^\top Z_k) Z_k^\top \mathbf{y}$.
- Further, a base learner is allowed to include a penalization defined by a matrix K_k which extends the estimation to $\hat{\boldsymbol{\theta}}_k = (Z_k^\top Z_k + K_k) Z_k^\top \mathbf{y}$.

Component-wise gradient boosting – Algorithm

Algorithm 2 Vanilla CWB algorithm

Input Train data $\mathcal{D}$, learning rate ν, number of boosting iterations M, loss function L, base learners $b_1, \ldots, b_K$

Output Model $\hat{f} = \hat{f}^{[M]}$

procedure CWB($\mathcal{D}, \nu, M, L, b_1, \ldots, b_K$)

  Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$

  while $m \leq M$ **do**

    $r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f = \hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$

    for $k \in \{1, \ldots, K\}$ **do**

      $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(Z_k^\top Z_k + K_k\right)^{-1} Z_k^\top r^{[m]}$

      $\text{SSE}_k = \sum_{i=1}^{n} (r^{[m](i)} - b_k(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}_k^{[m]}))^2$

    $k^{[m]} = \arg\min_{k \in \{1,\ldots,K\}} \text{SSE}_k$

    $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu b_{k^{[m]}}(\mathbf{x}|\hat{\boldsymbol{\theta}}_{k^{[m]}}^{[m]})$

  return $\hat{f} = \hat{f}^{[M]}$

Component-wise gradient boosting – Example

Example throughout this presentation is a subset of a WHO data set[1] about life expectation in years per country:

Life.expectancy	Country	Year	BMI	Adult.Mortality
53.2	ETH	2002	12.9	369
81.0	GER	2015	62.3	68
76.8	USA	2000	6.1	114
79.1	USA	2014	69.1	14
58.9	ZAF	2011	47.9	413
69.0	ZAF	2013	49.5	371

[1] Full description and data is available at
`kaggle.com/datasets/kumarajarshi/life-expectancy-who`

Component-wise gradient boosting – Example

Example throughout this presentation is a subset of a WHO data set[1] about life expectation in years per country:

- Target variable is `Life.expectancy` in years.
- Features are `Country`, `Year`, `Alcohol` recorded per capital (15+) consumption (in liters of pure alcohol), and `Adult.Mortality` rates of both sexes of dying between 15 and 60 years per 1000 population.
- Numerical features `Year`, `Alcohol` and `Adult.Mortality` are modeled as P-splines (Eilers and Marx, 1996) and `Country` as one-hot-encoded linear model with ridge penalty.

[1] Full description and data is available at
`kaggle.com/datasets/kumarajarshi/life-expectancy-who`

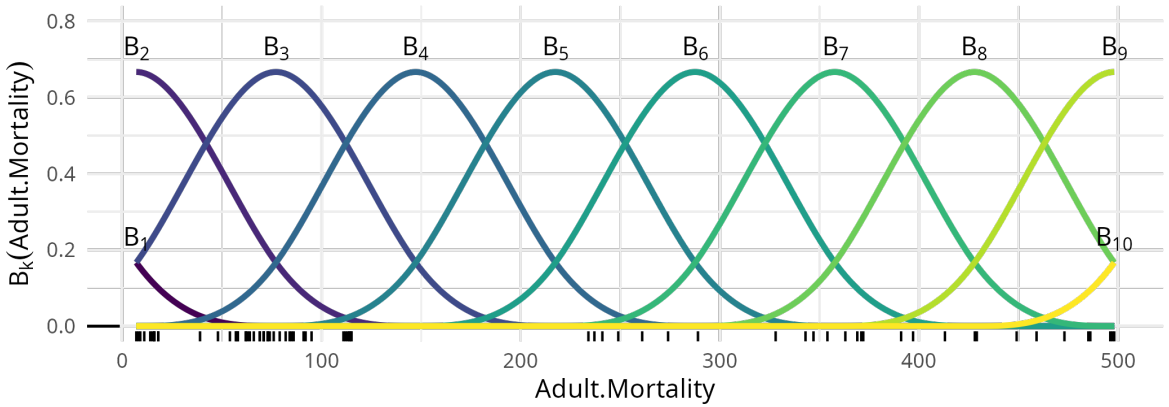
Component-wise gradient boosting

Base learner

B/P-Spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

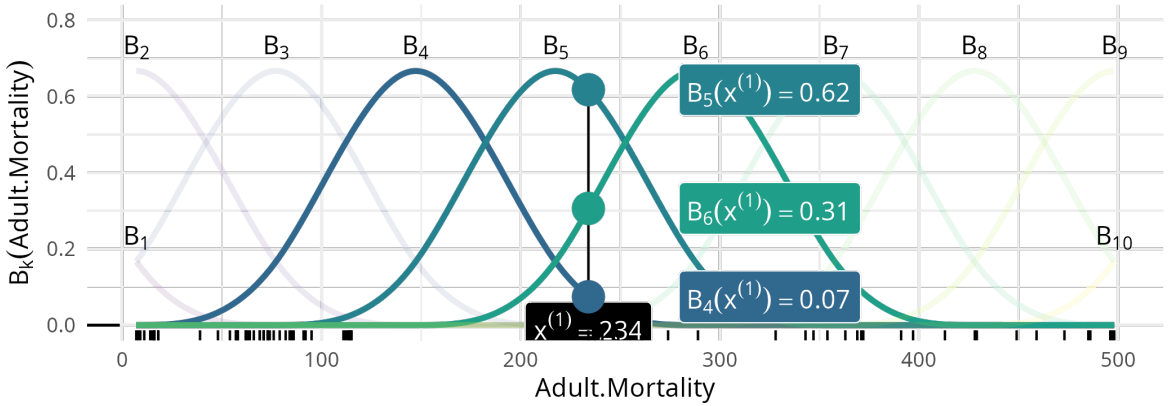


$$Z_k = \begin{pmatrix} g_k^\top(\mathbf{x}^{(1)}) \\ \vdots \\ g_k^\top(\mathbf{x}^{(n)}) \end{pmatrix} = \begin{pmatrix} B_{k,1}(\mathbf{x}^{(1)}) & \ldots & B_{k,d_k}(\mathbf{x}^{(1)}) \\ \vdots & & \vdots \\ B_{k,1}(\mathbf{x}^{(n)}) & \ldots & B_{k,d_k}(\mathbf{x}^{(n)}) \end{pmatrix} \in \mathbb{R}^{n \times d_k}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

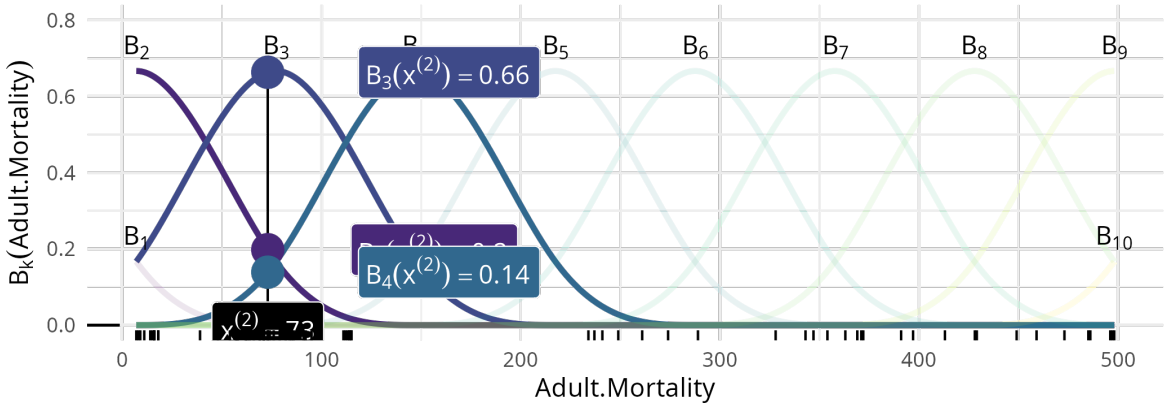


$$Z_k = \begin{array}{c} \begin{array}{cccccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \end{array} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix} \end{array}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

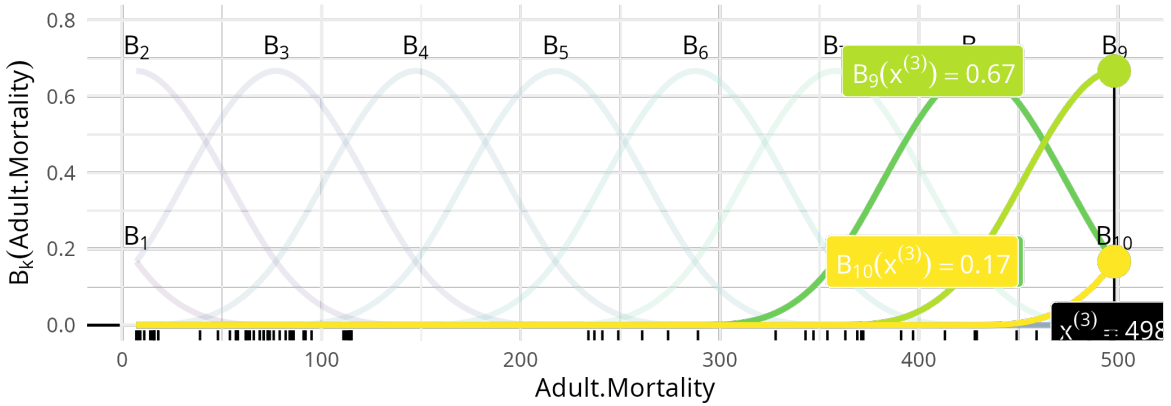


$$Z_k = \begin{pmatrix} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \\ 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ & & & & & & & & & \\ & & & & & & & & & \end{pmatrix}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

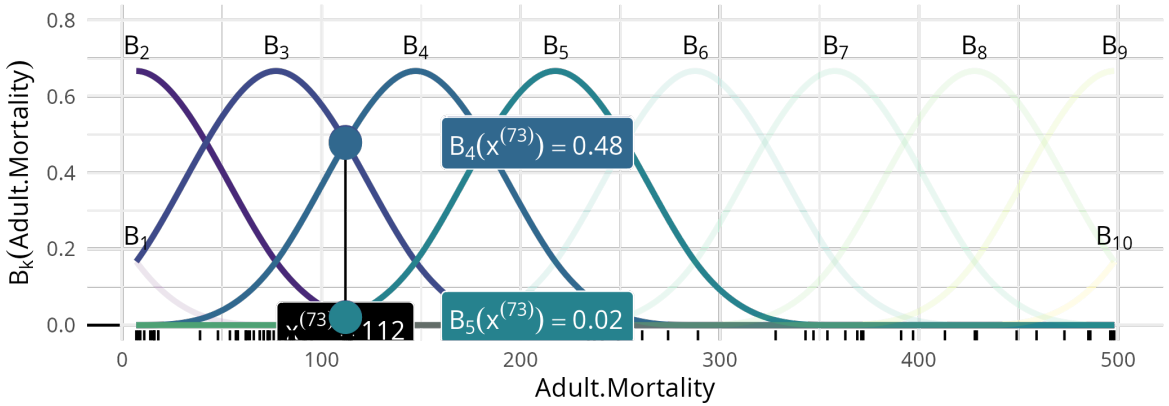


$$Z_k = \begin{array}{c} \begin{array}{cccccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \end{array} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \end{pmatrix} \end{array}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

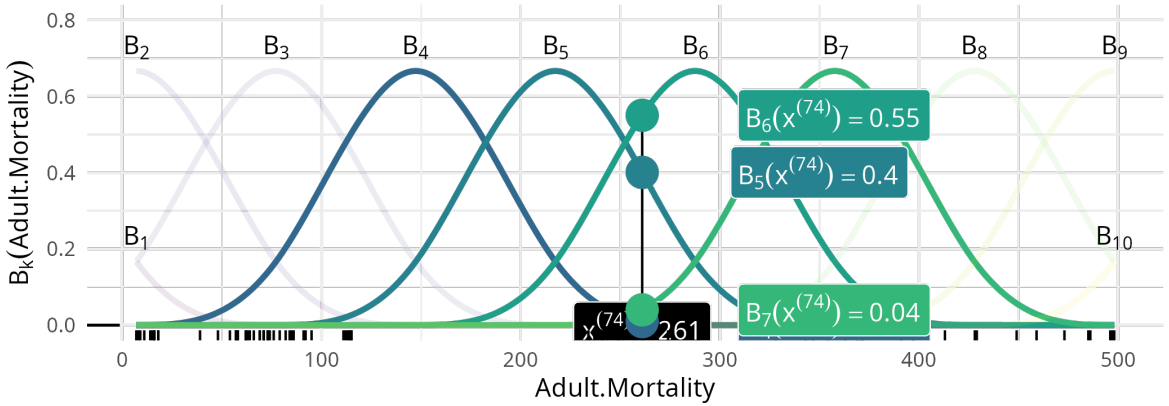


$$Z_k = \begin{pmatrix} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \\ 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).

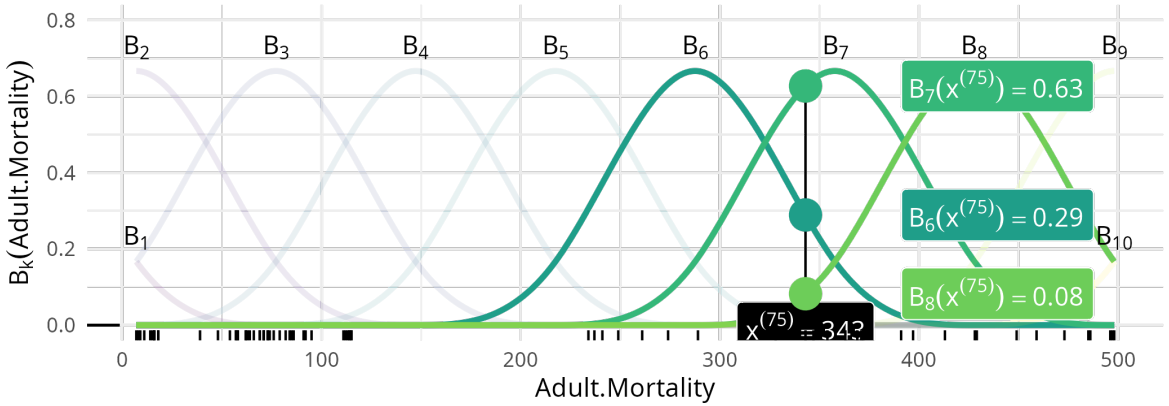


$$Z_k = \begin{pmatrix} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \\ 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.40 & 0.55 & 0.04 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

B/P-spline base learner

$$g_k(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$$

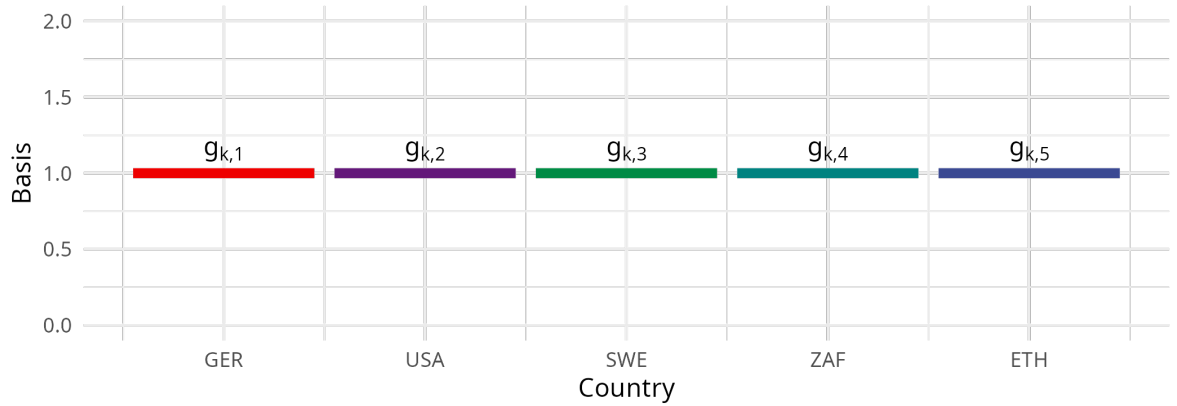
B-spline basis B of a pre-defined degree (Eilers and Marx, 1996).



$$Z_k = \begin{pmatrix} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 & B_9 & B_{10} \\ 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.40 & 0.55 & 0.04 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.63 & 0.08 & 0.00 & 0.00 \end{pmatrix}$$

Categorical base learner

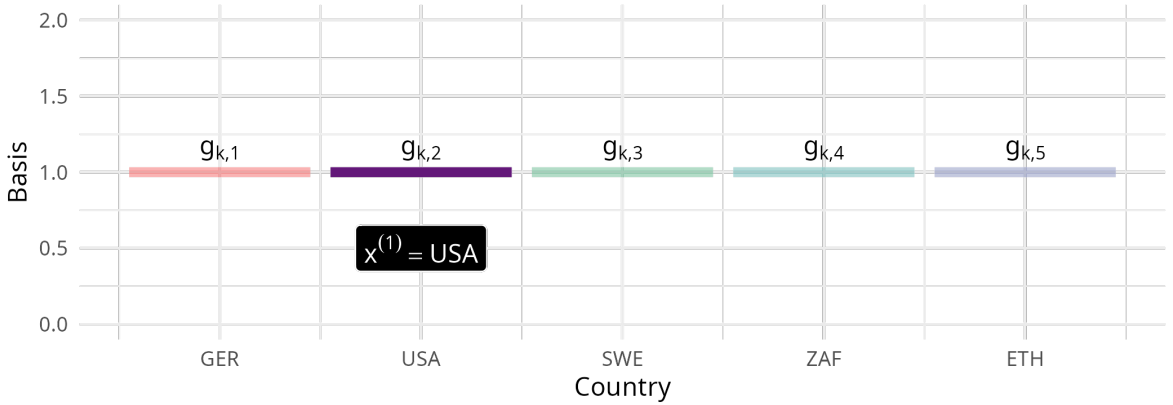
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{pmatrix} g_k^\top(\mathbf{x}^{(1)}) \\ \vdots \\ g_k^\top(\mathbf{x}^{(n)}) \end{pmatrix} = \begin{pmatrix} \mathbb{1}_{\{\mathbf{x}^{(1)}=1\}} & \cdots & \mathbb{1}_{\{\mathbf{x}^{(1)}=G\}} \\ \vdots & & \vdots \\ \mathbb{1}_{\{\mathbf{x}^{(n)}=1\}} & \cdots & \mathbb{1}_{\{\mathbf{x}^{(n)}=G\}} \end{pmatrix} \in \mathbb{R}^{n \times G}$$

Categorical base learner

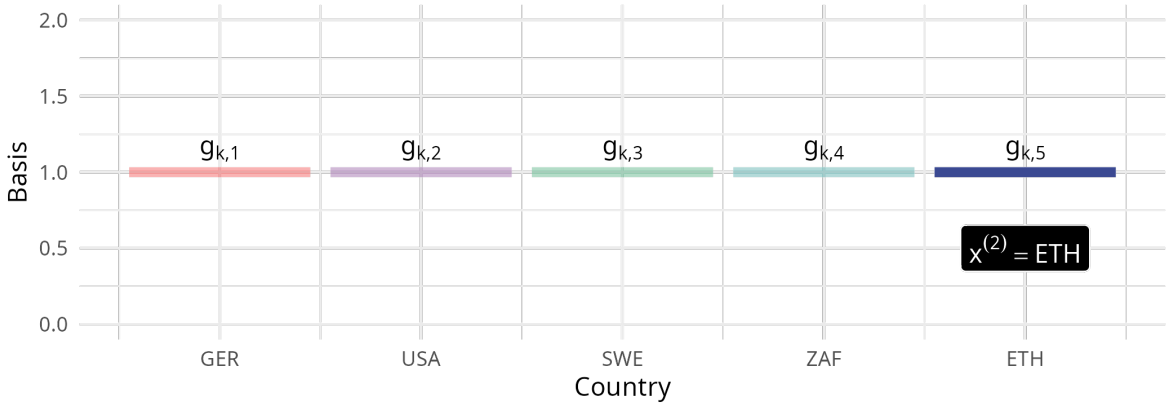
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Categorical base learner

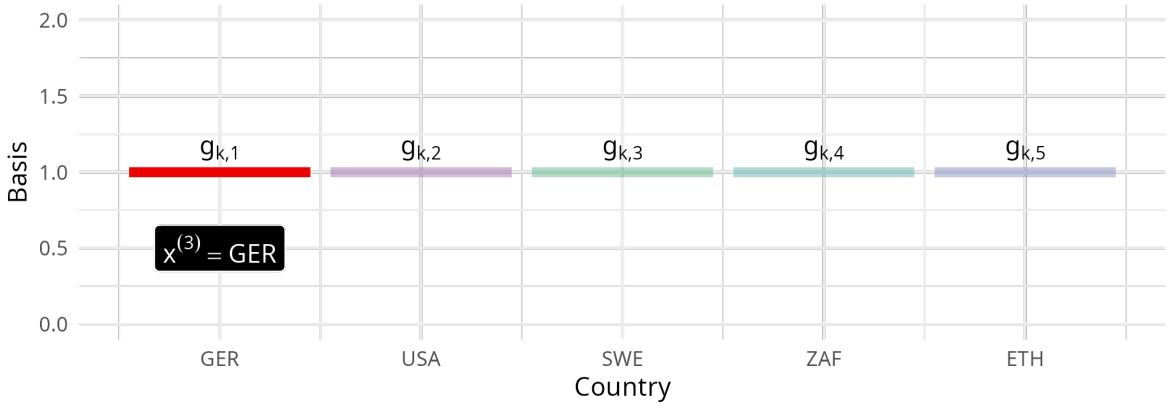
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{matrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ & & & & \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

Categorical base learner

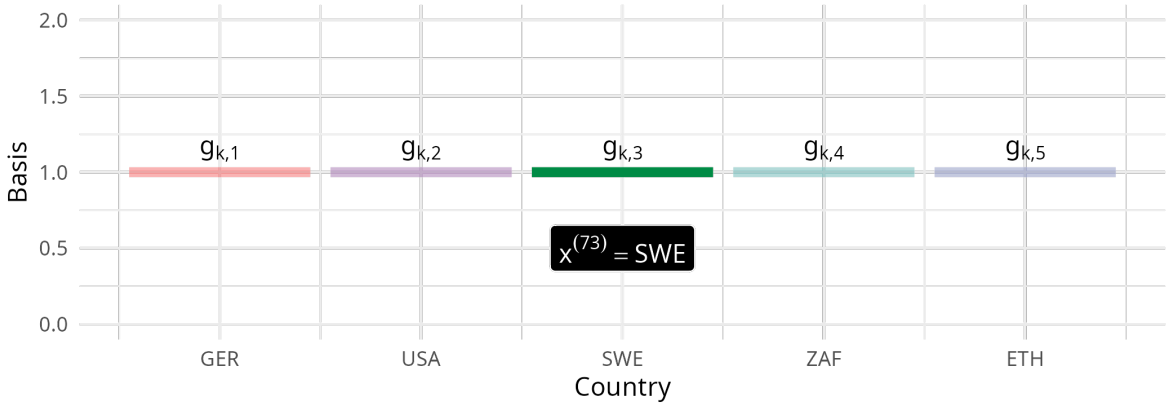
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

Categorical base learner

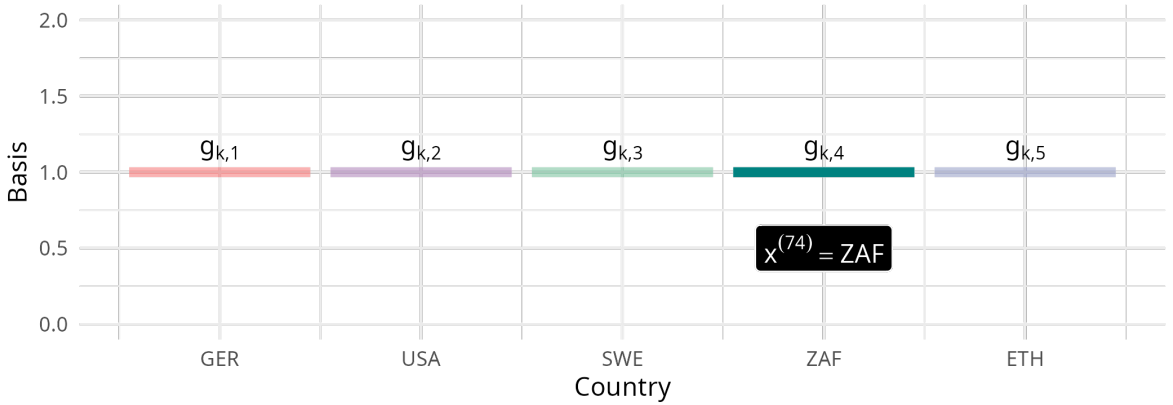
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{matrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Categorical base learner

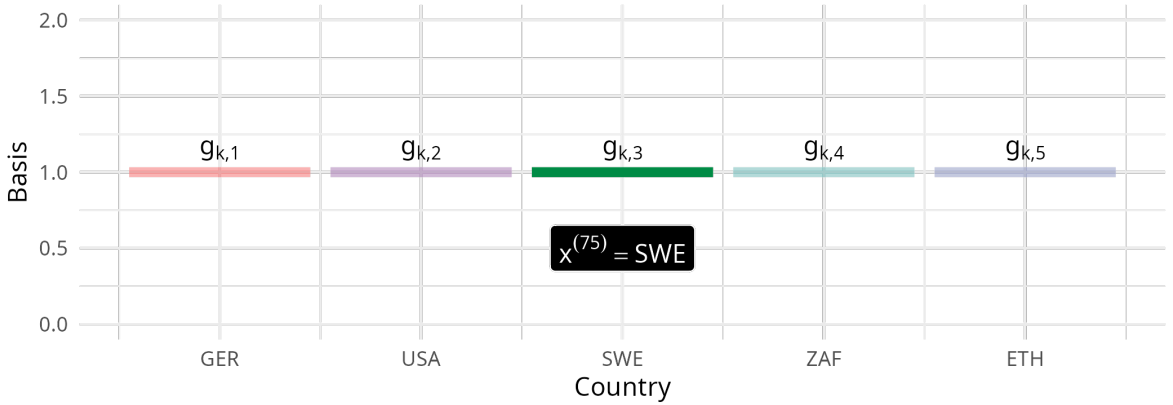
$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{matrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Categorical base learner

$$g_k(x) = (g_{k,1}(x), \ldots, g_{k,G}(x))^\top = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top, \quad x \in \{1, \ldots, G\}$$



$$Z_k = \begin{array}{c} \begin{matrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

(Row-wise) tensor product base learner

Combination (interaction) $b_k \odot b_l$ between to base learners b_k and b_l:

$$g_k(\mathbf{x}) \otimes g_l(\mathbf{x}) = (g_{k,1}(\mathbf{x})g_l(\mathbf{x})^\top, \ldots, g_{k,d_k}(\mathbf{x})g_l(\mathbf{x})^\top)^\top$$

Design matrix:

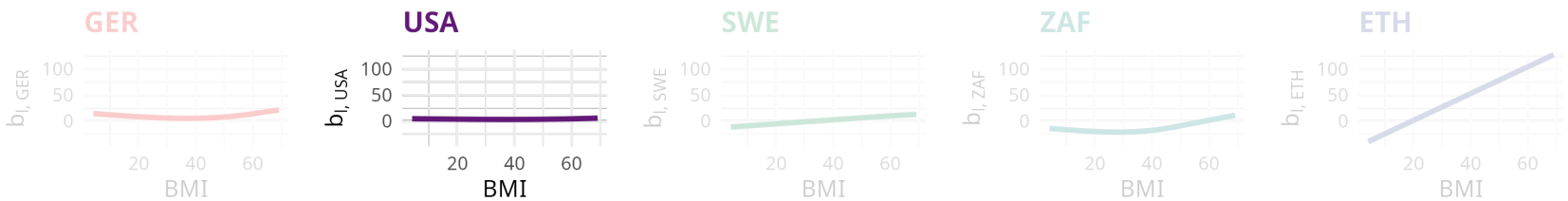
$$Z_k \odot Z_l = \begin{pmatrix} (g_k(\mathbf{x}^{(1)}) \otimes g_l(\mathbf{x}^{(1)}))^\top \\ \vdots \\ (g_k(\mathbf{x}^{(n)}) \otimes g_l(\mathbf{x}^{(n)}))^\top \end{pmatrix} \in \mathbb{R}^{n \times d_k d_l}$$

(Row-wise) tensor product base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{pmatrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \\ 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ & & & & \end{pmatrix}$$

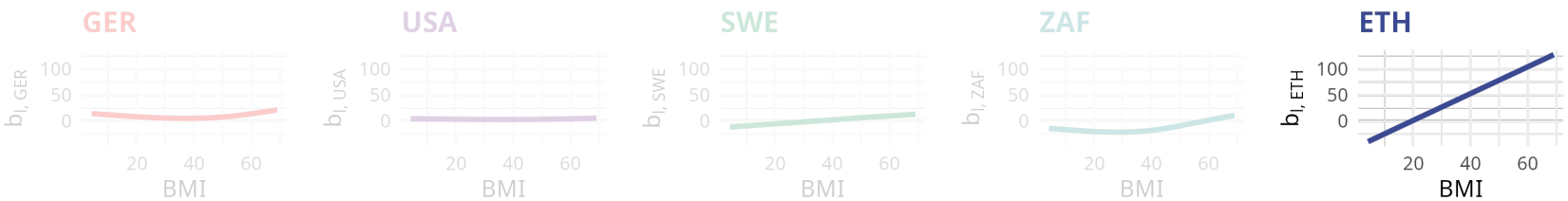


(Row-wise) tensor product base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \end{pmatrix} \end{array}$$

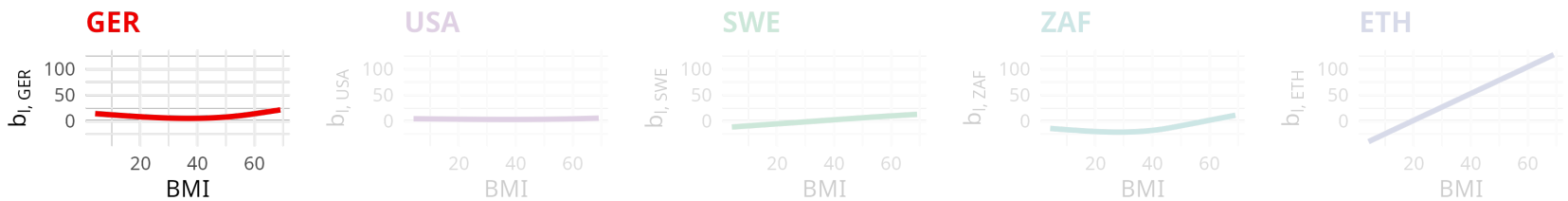


(Row-wise) tensor product base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{array}{c} \begin{array}{ccccc} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{array} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

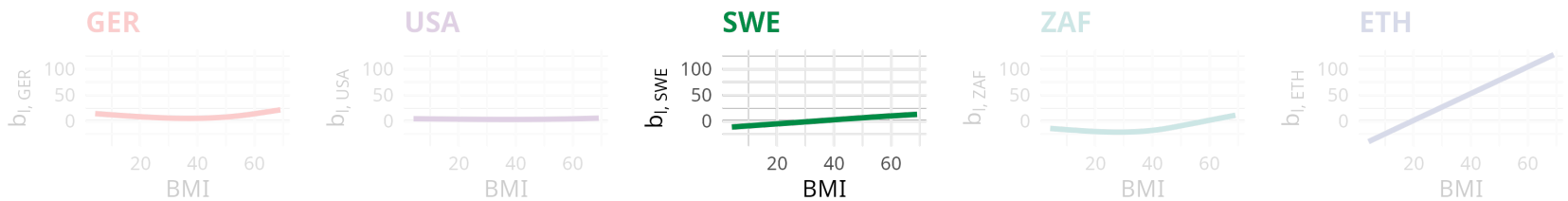


(Row-wise) tensor product base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{pmatrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \\ 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & g_l(x^{(73)}) & 0 & 0 \end{pmatrix}$$

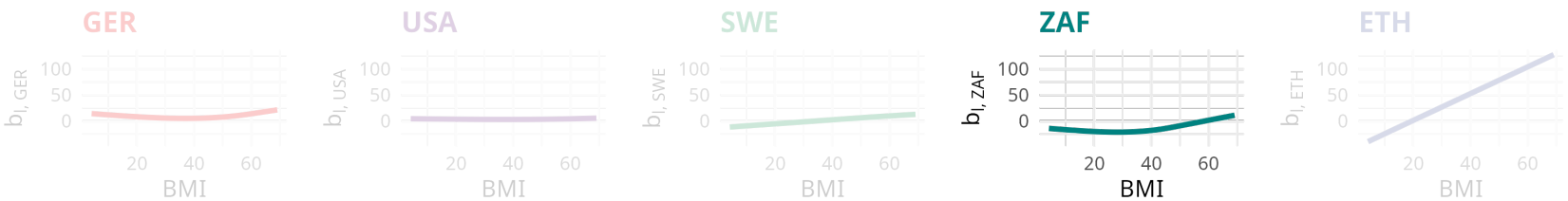


(Row-wise) tensor product base learner

Example:

- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{pmatrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \\ 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & g_l(x^{(73)}) & 0 & 0 \\ 0 & 0 & 0 & g_l(x^{(74)}) & 0 \end{pmatrix}$$

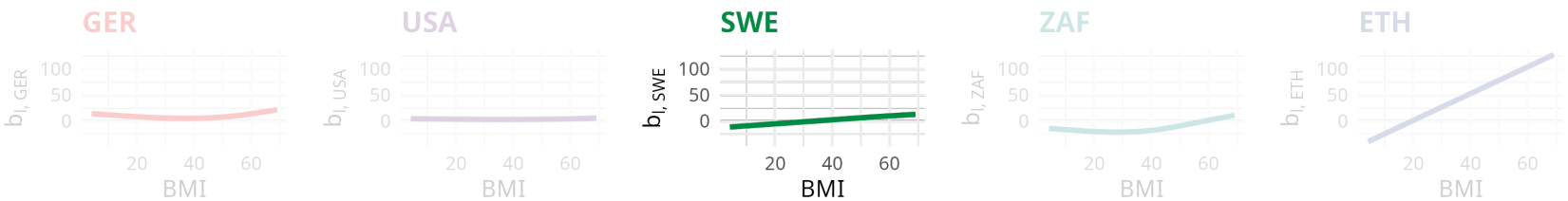


(Row-wise) tensor product base learner

Example:

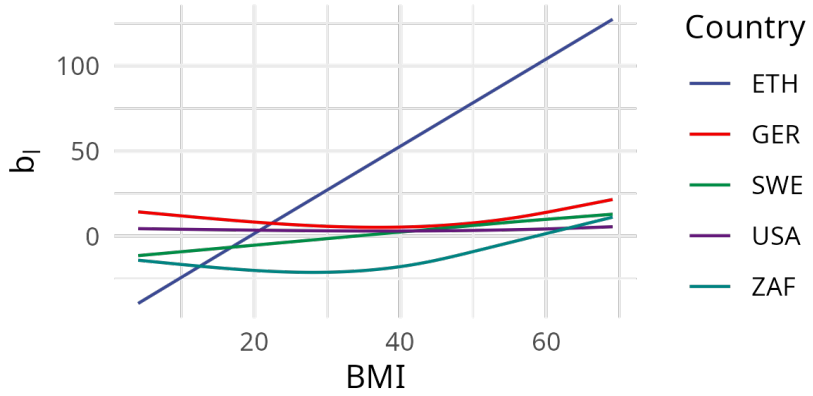
- b_k encodes the country: $g_k(x) = (\mathbb{1}_{\{x=1\}}, \ldots, \mathbb{1}_{\{x=G\}})^\top$
- b_l uses a B-spline basis for BMI: $g_l(x) = (B_{k,1}(x), \ldots, B_{k,d_k}(x))^\top$

$$Z_k \odot Z_l = \begin{array}{c} \begin{matrix} g_{k,GER} & g_{k,USA} & g_{k,SWE} & g_{k,ZAF} & g_{k,ETH} \end{matrix} \\ \begin{pmatrix} 0 & g_l(x^{(1)}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_l(x^{(2)}) \\ g_l(x^{(3)}) & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & g_l(x^{(73)}) & 0 & 0 \\ 0 & 0 & 0 & g_l(x^{(74)}) & 0 \\ 0 & 0 & g_l(x^{(75)}) & 0 & 0 \end{pmatrix} \end{array}$$

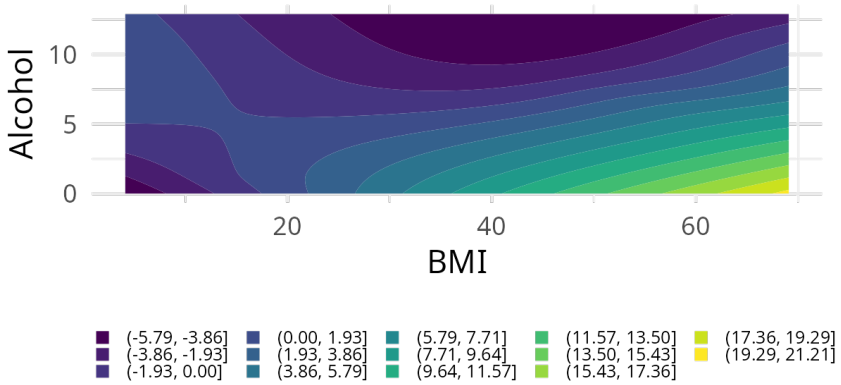


(Row-wise) tensor product base learner

- Categoric - numeric base learner combination:



- Numeric - numeric base learner combination:



Component-wise gradient boosting

Fitting process

Initialization vs. fitting phase

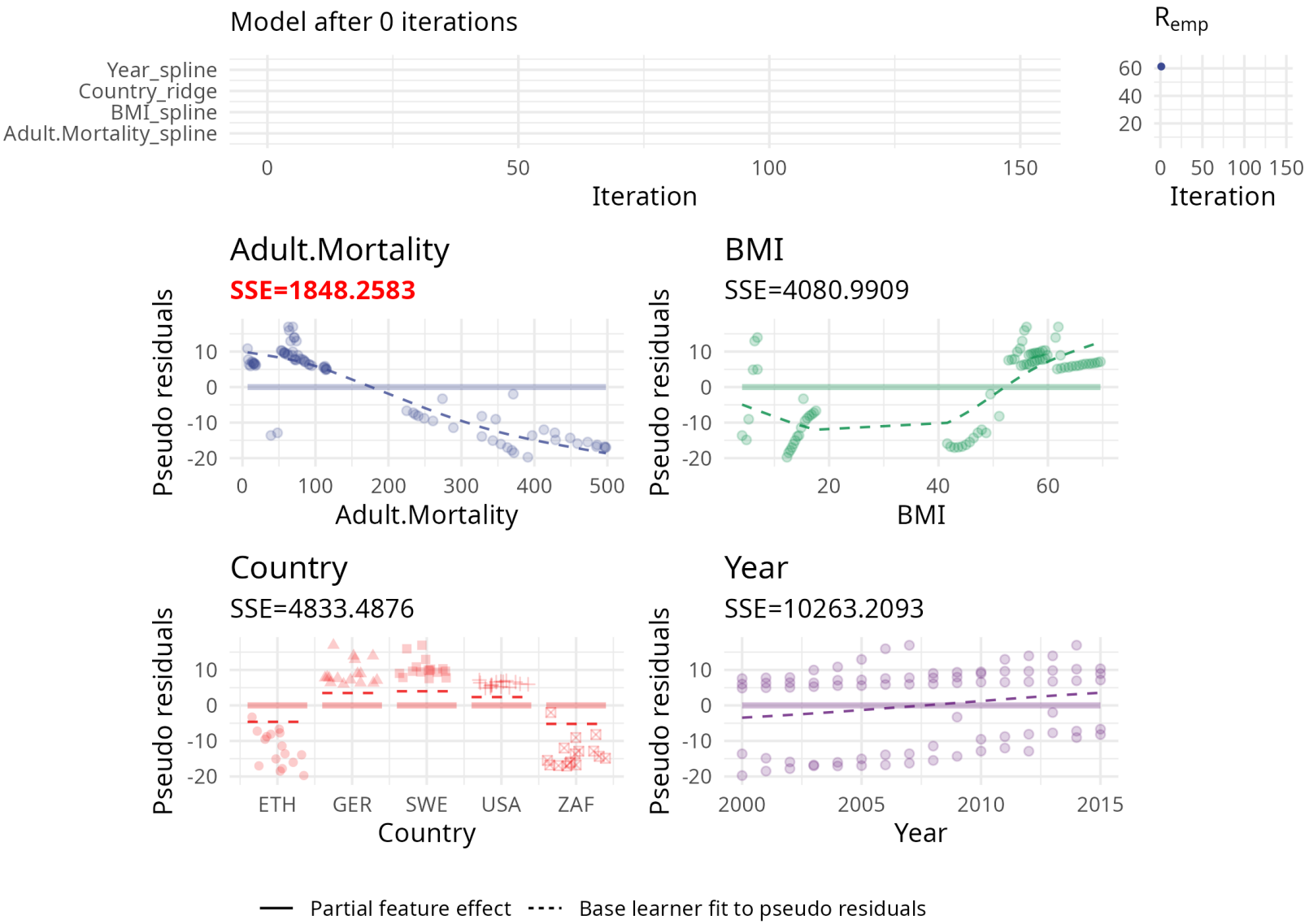
Initialization:

- All design matrices Z_k and penalties K_k are calculated for $k = 1, \ldots, K$ and stored.
- Additionally, the Cholesky decomposition L_k of $Z_k^\top Z_k + K_k$ is calculated and stored.

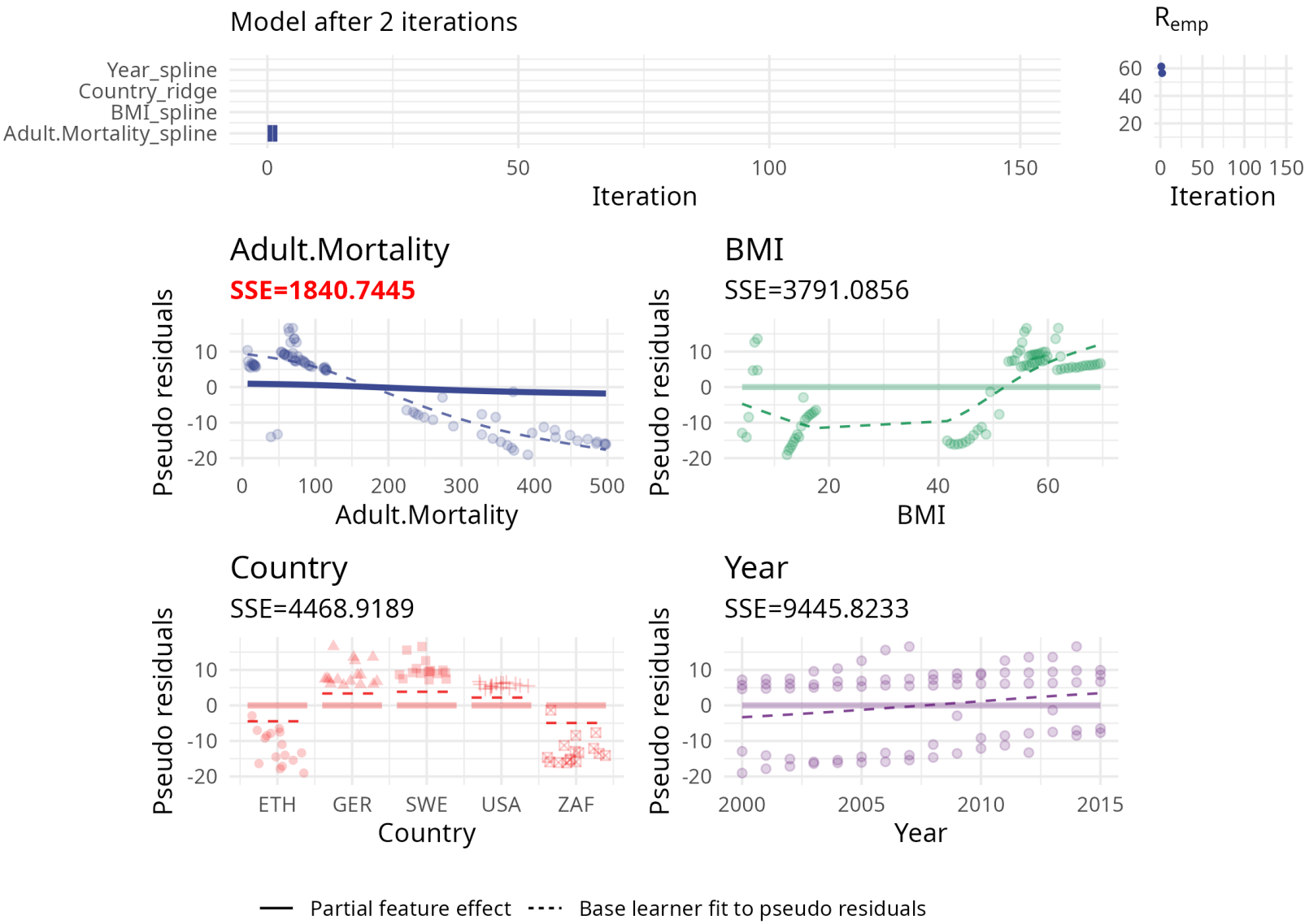
Fitting:

- Algorithm 3 is executed.
- But, instead of calculating $\hat{\boldsymbol{\theta}}_k^{[m]} = \left(Z_k^\top Z_k + K_k\right)^{-1} Z_k^\top \boldsymbol{r}^{[m]}$ in each iteration, reuse L_k to calculate $\hat{\boldsymbol{\theta}}^{[m]}$.

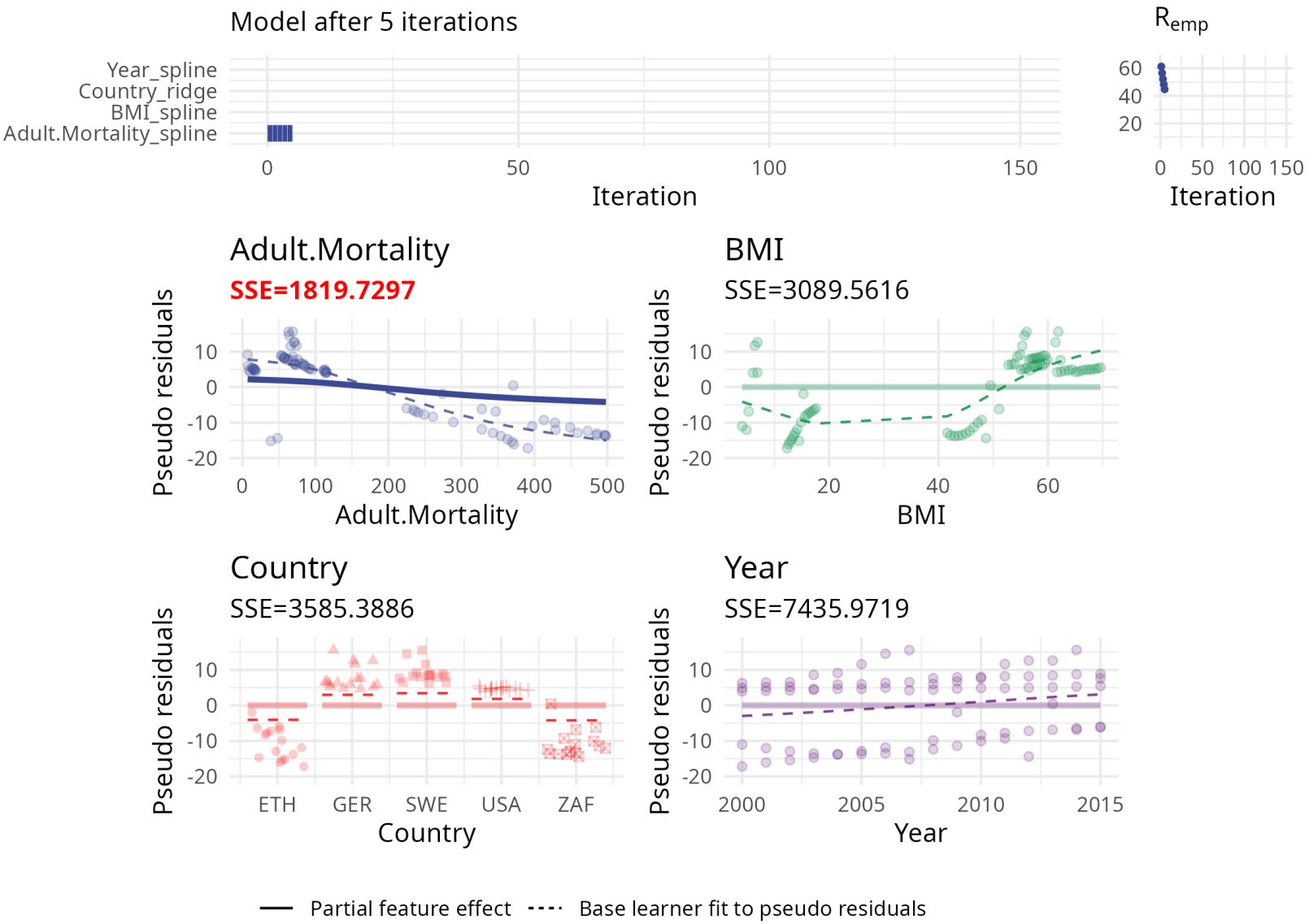
Fitting process: Life expectancy



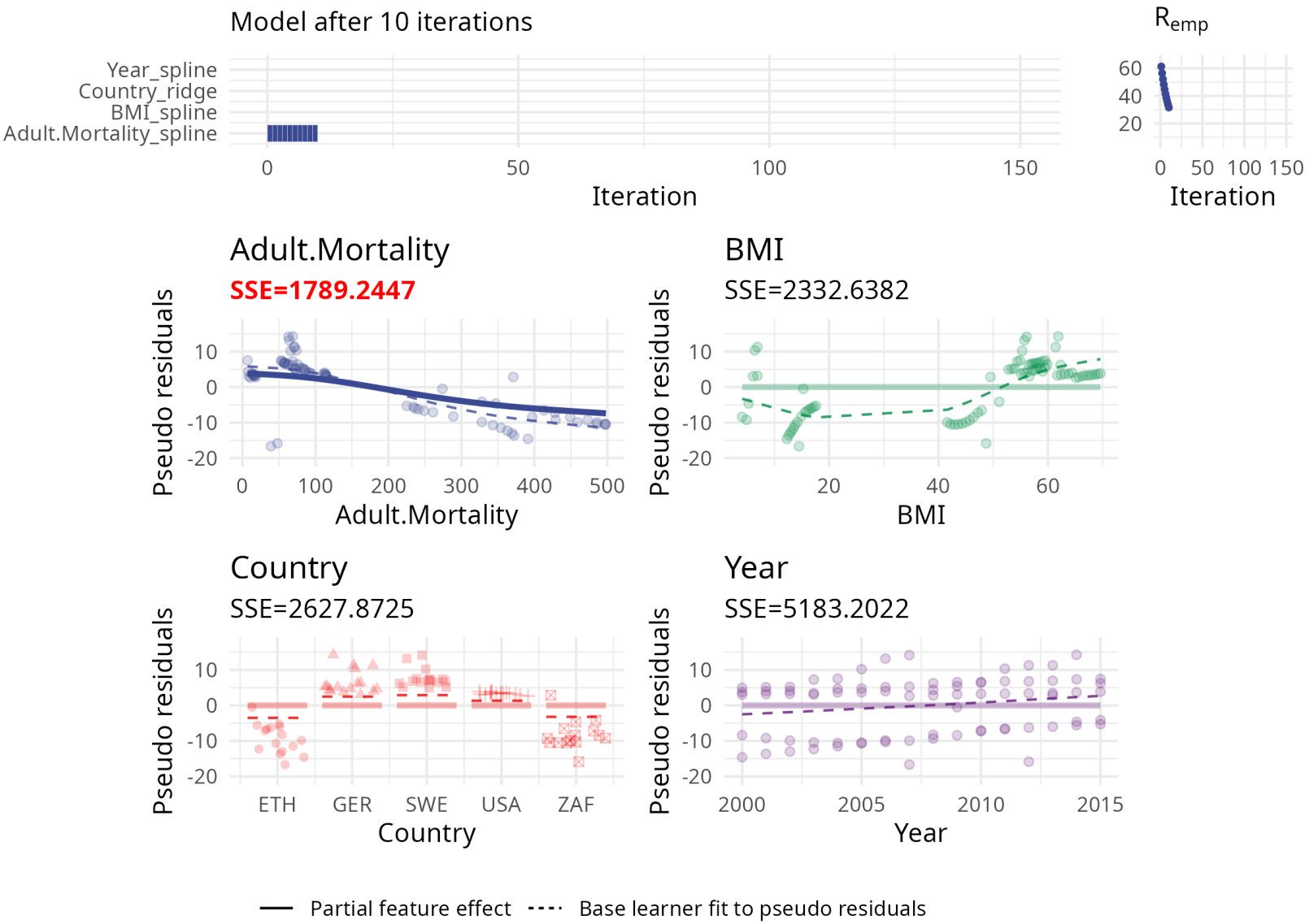
Fitting process: Life expectancy



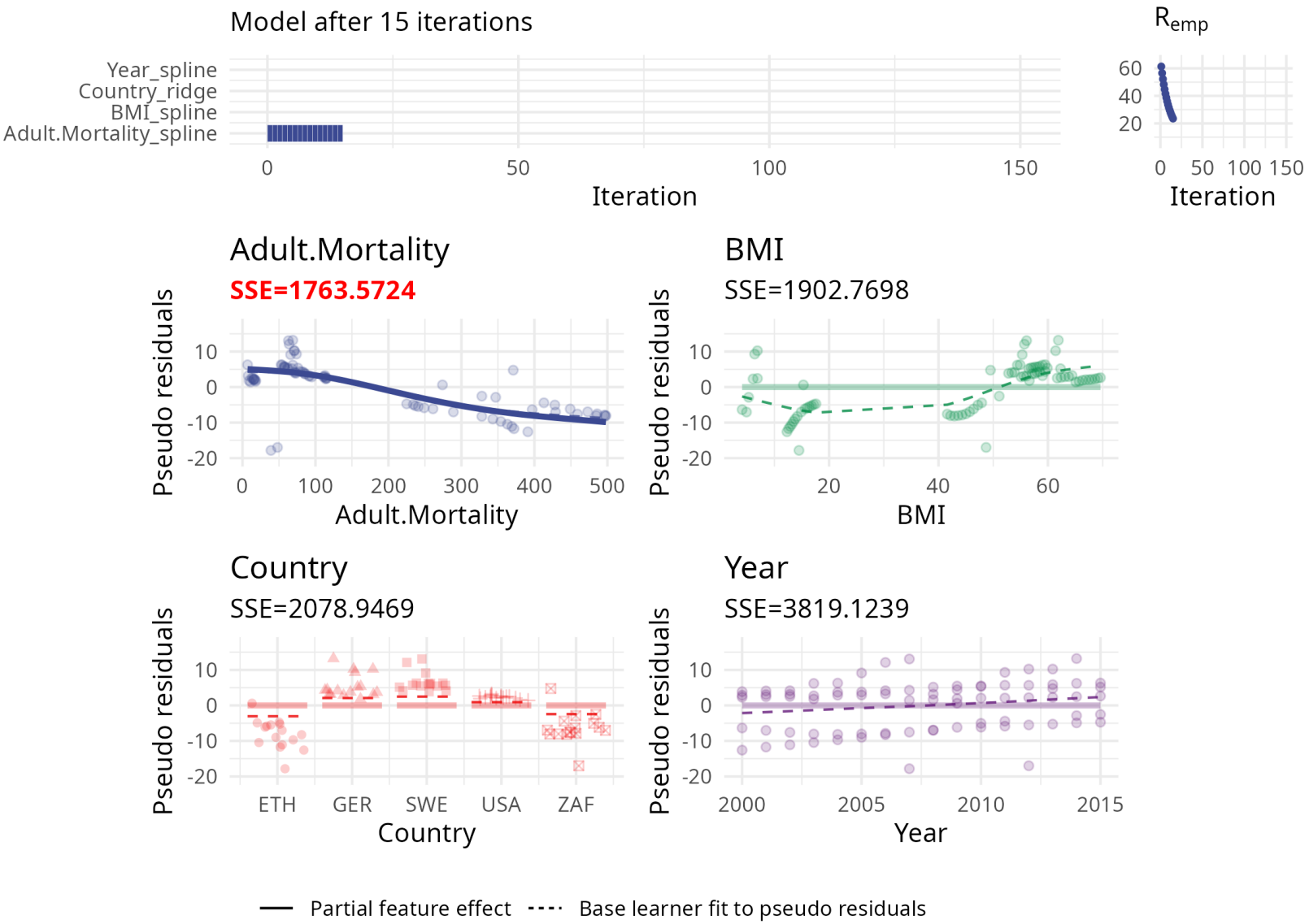
Fitting process: Life expectancy



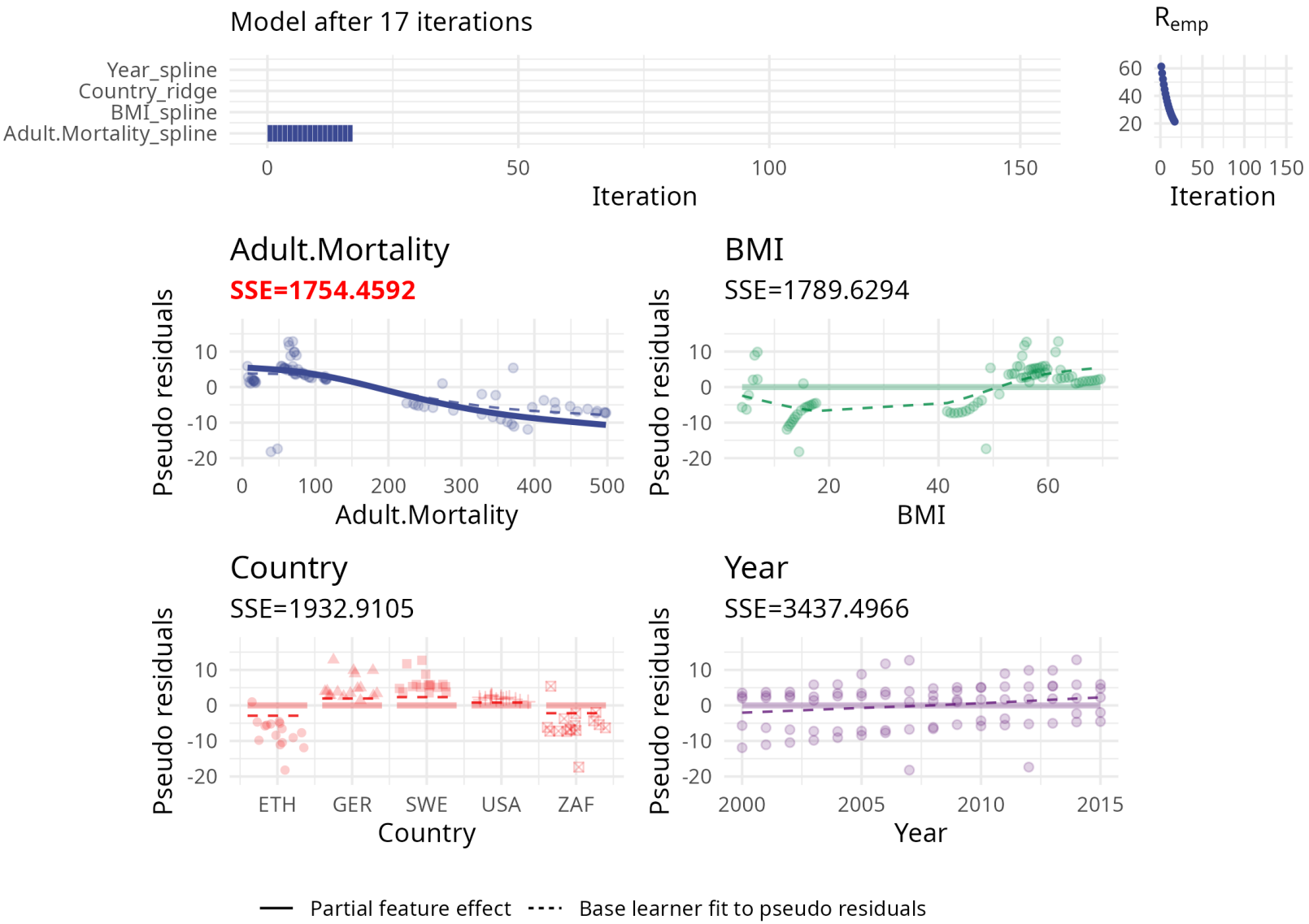
Fitting process: Life expectancy



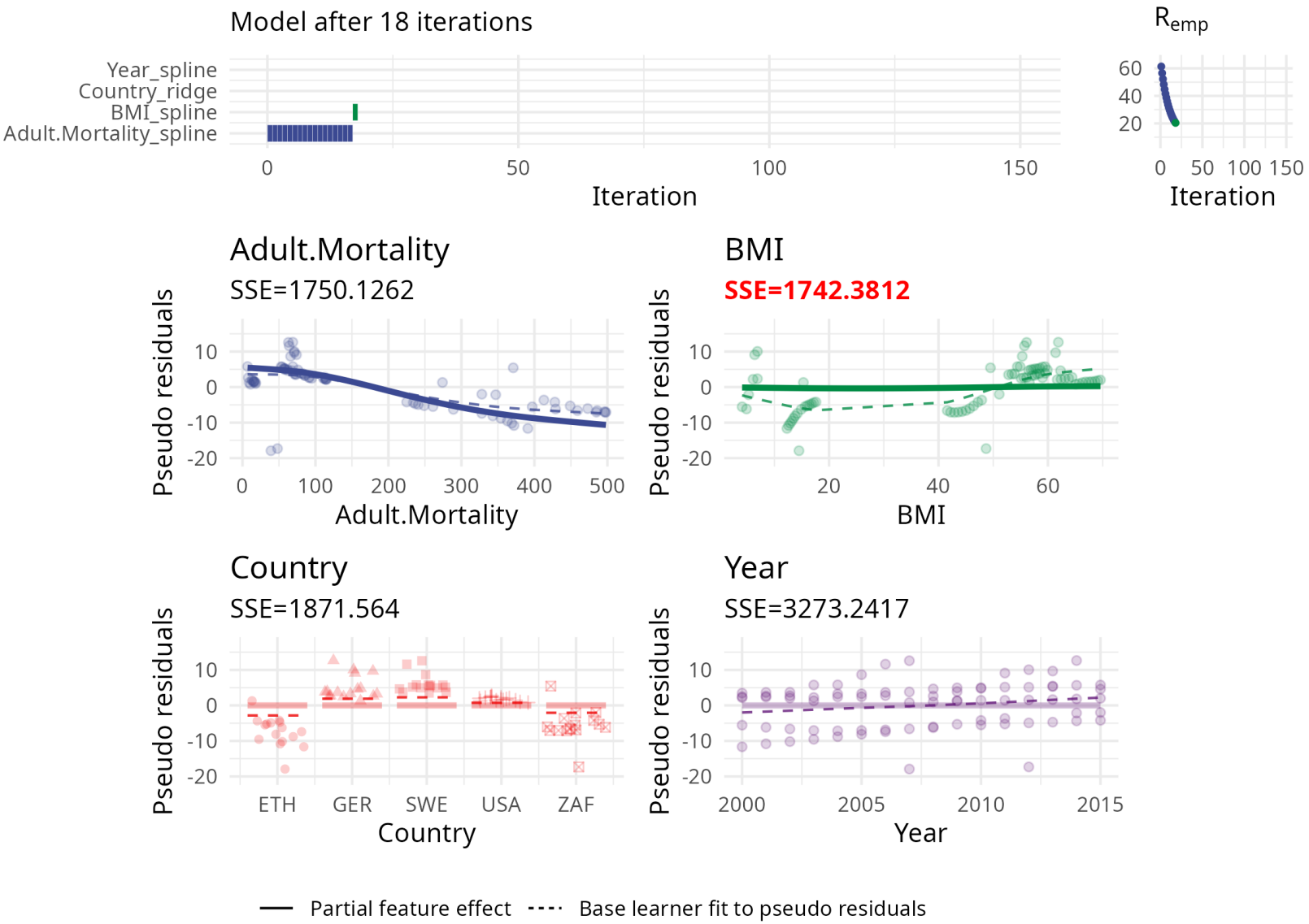
Fitting process: Life expectancy



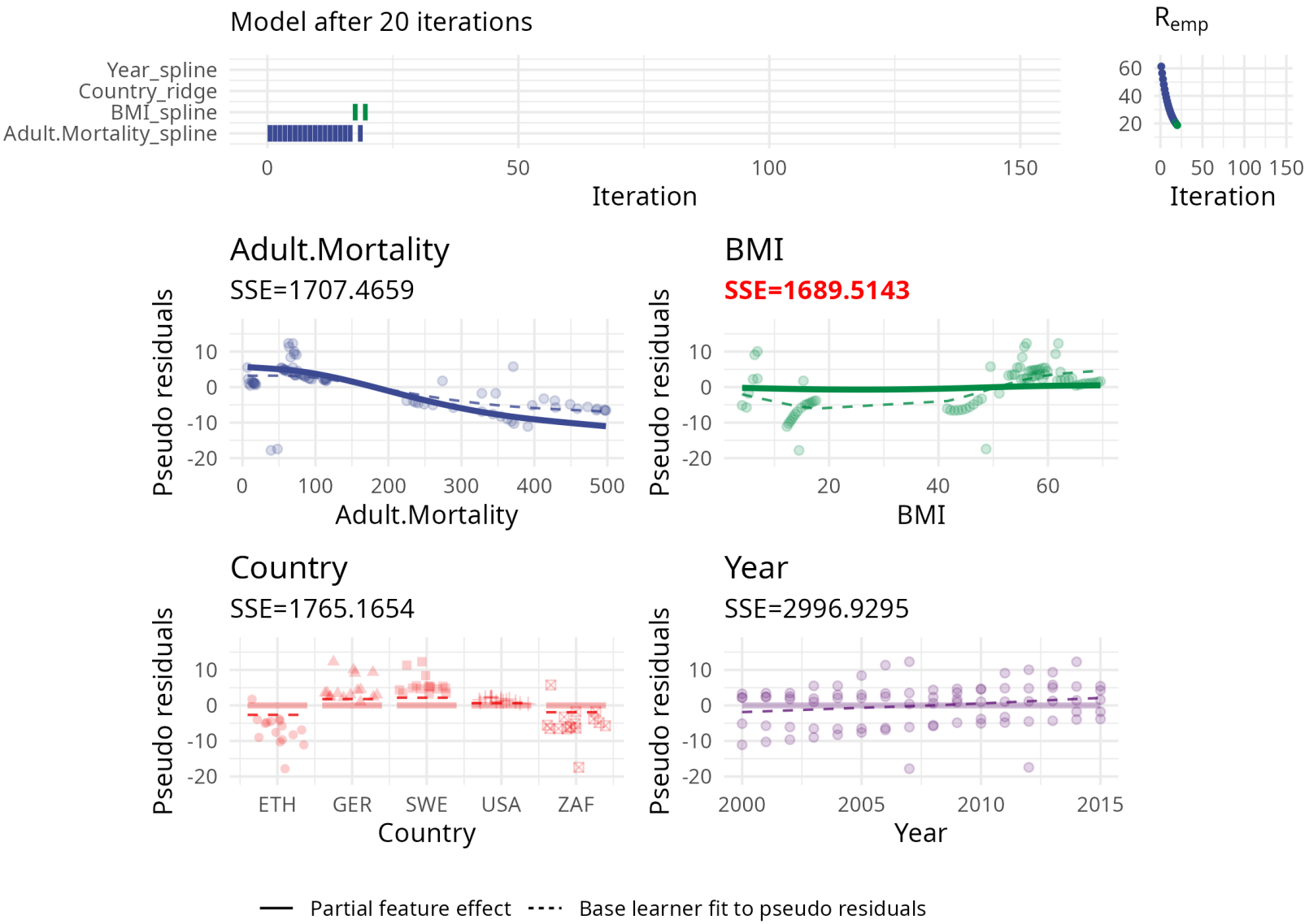
Fitting process: Life expectancy



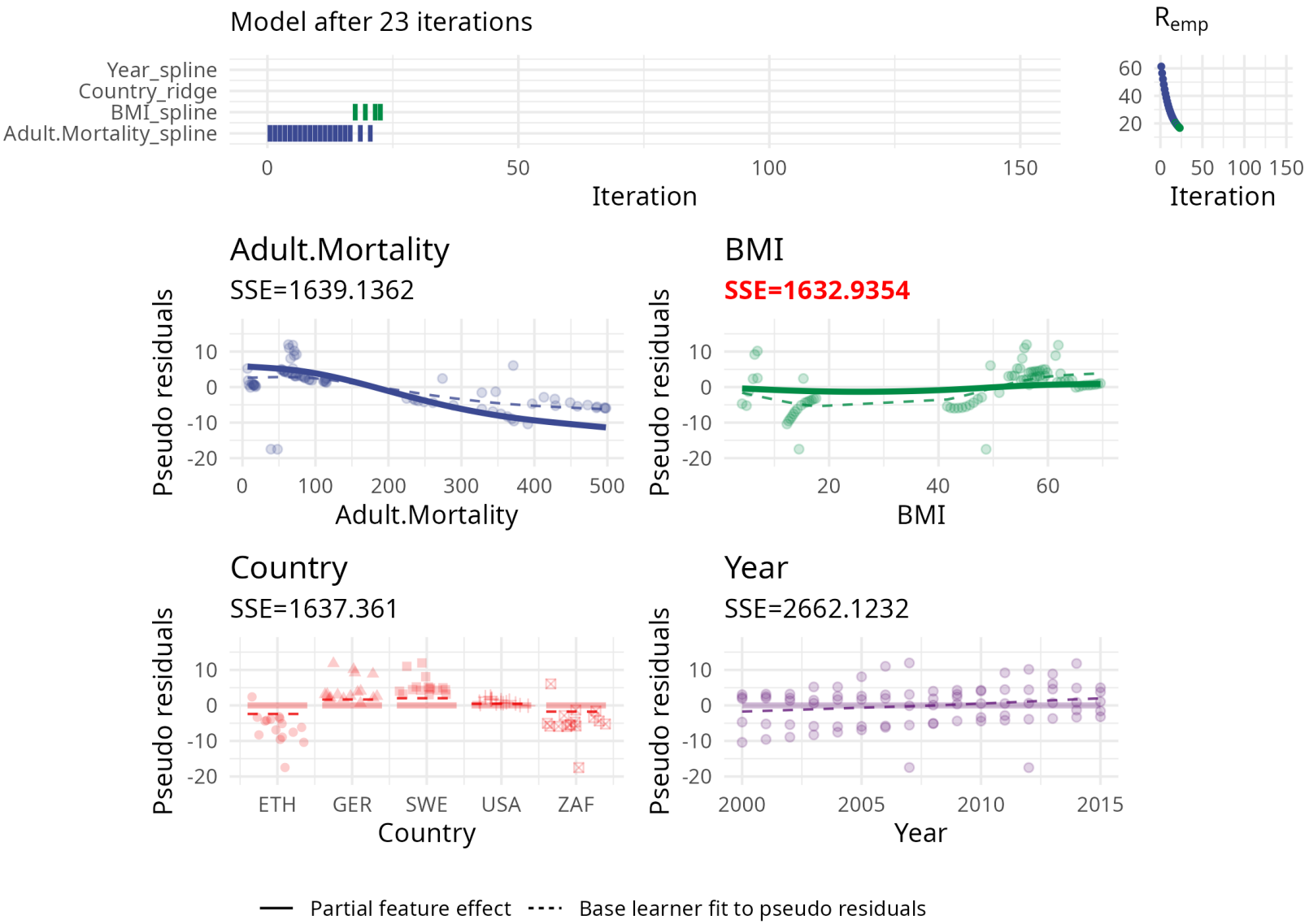
Fitting process: Life expectancy



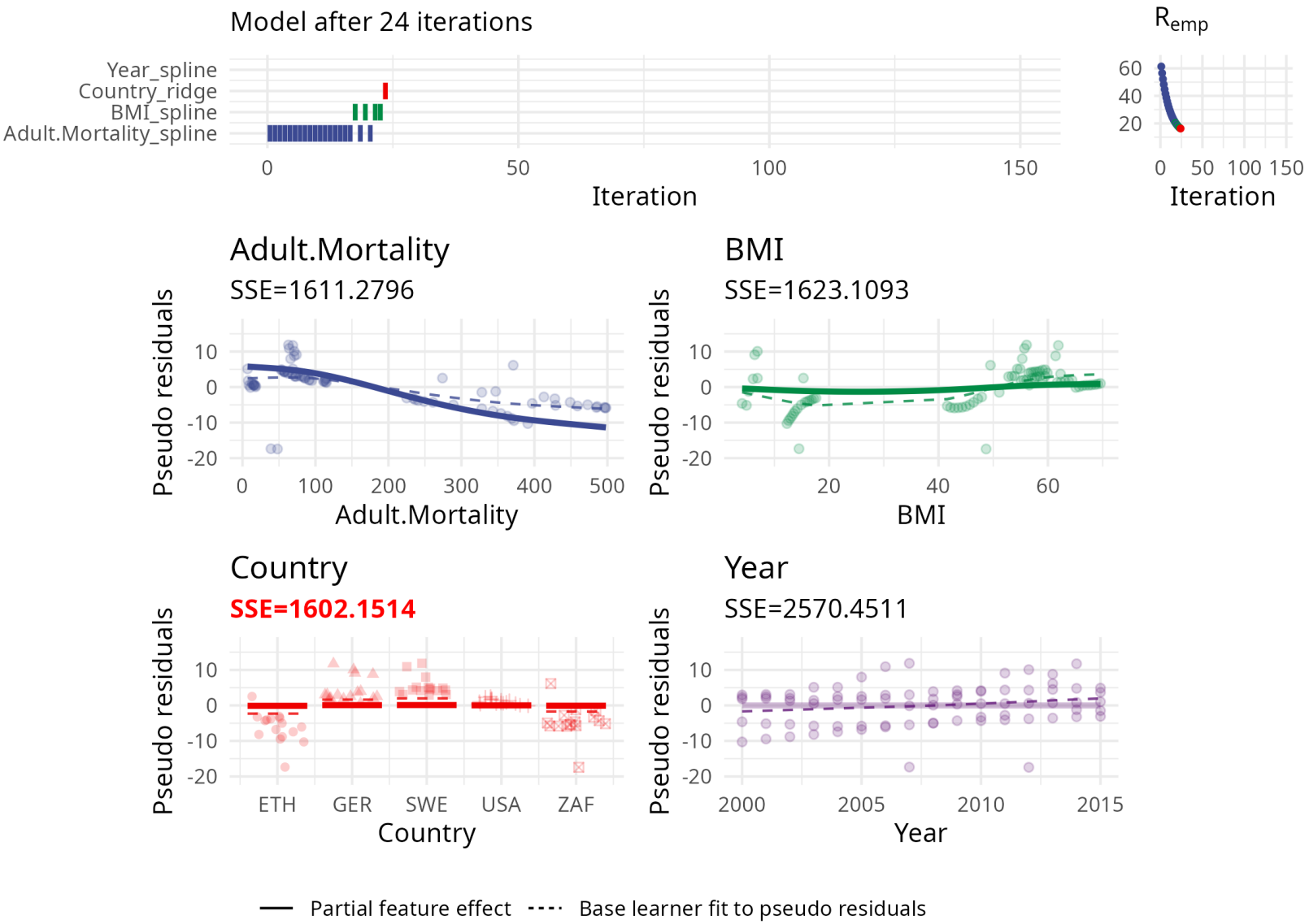
Fitting process: Life expectancy



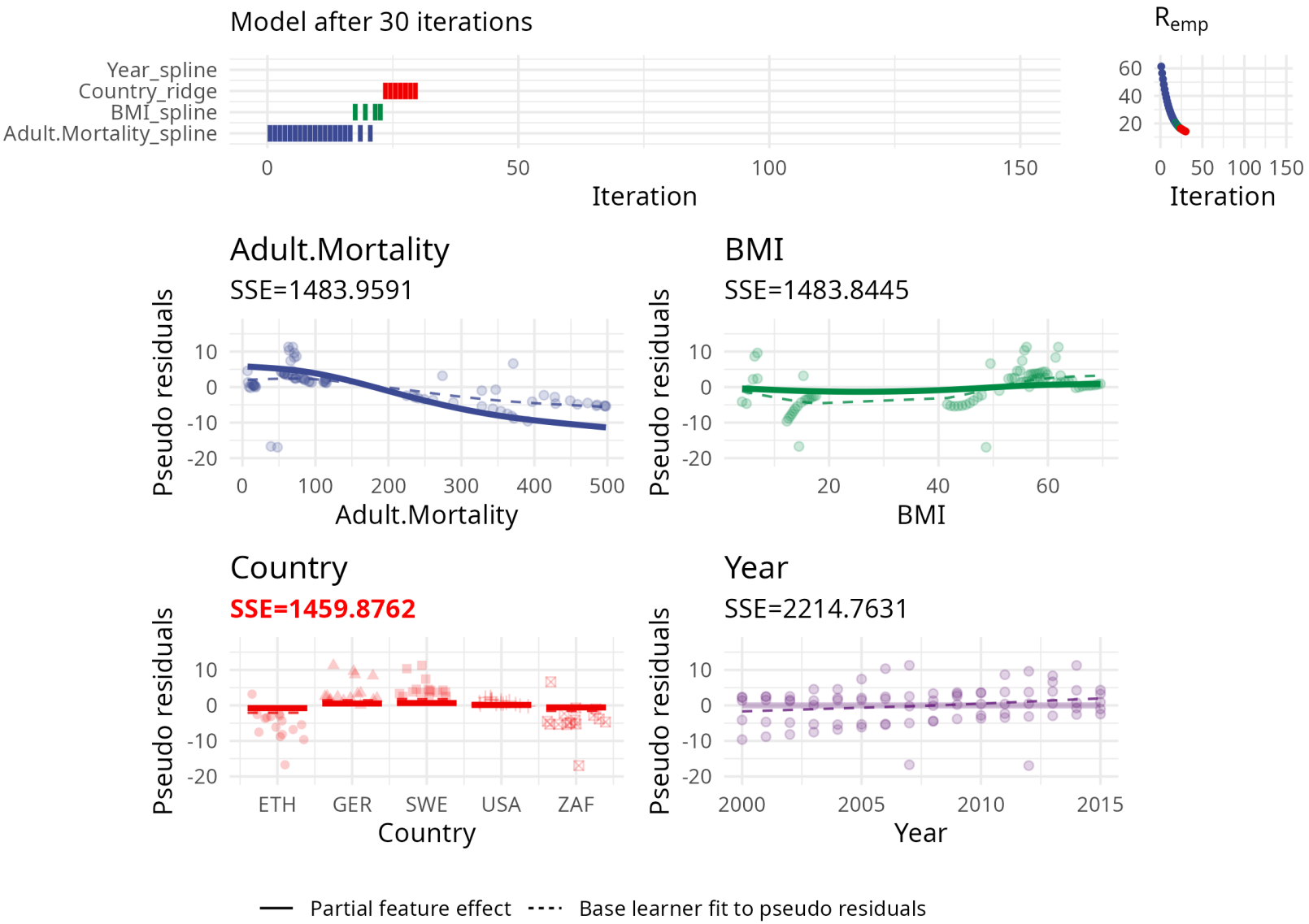
Fitting process: Life expectancy



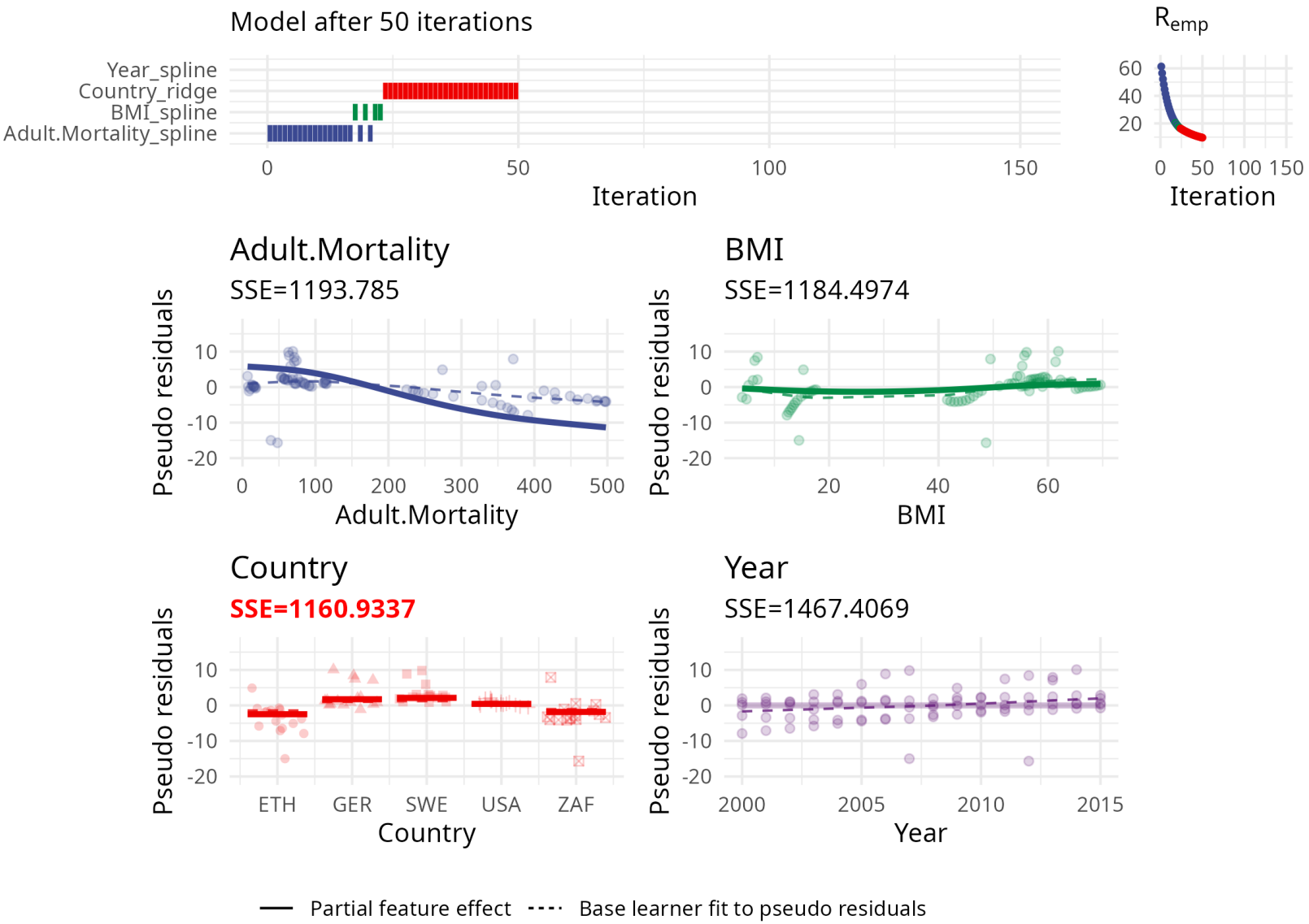
Fitting process: Life expectancy



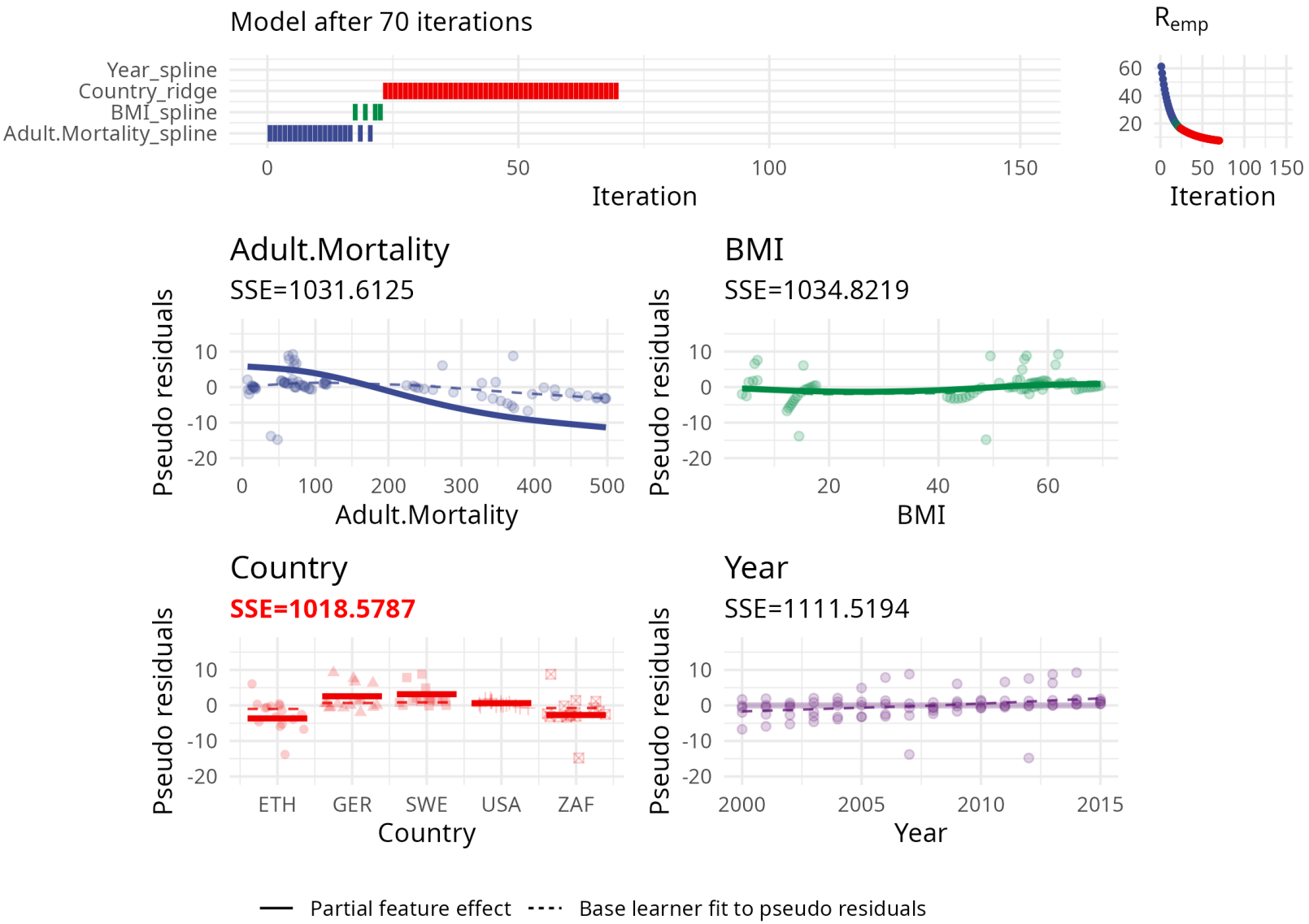
Fitting process: Life expectancy



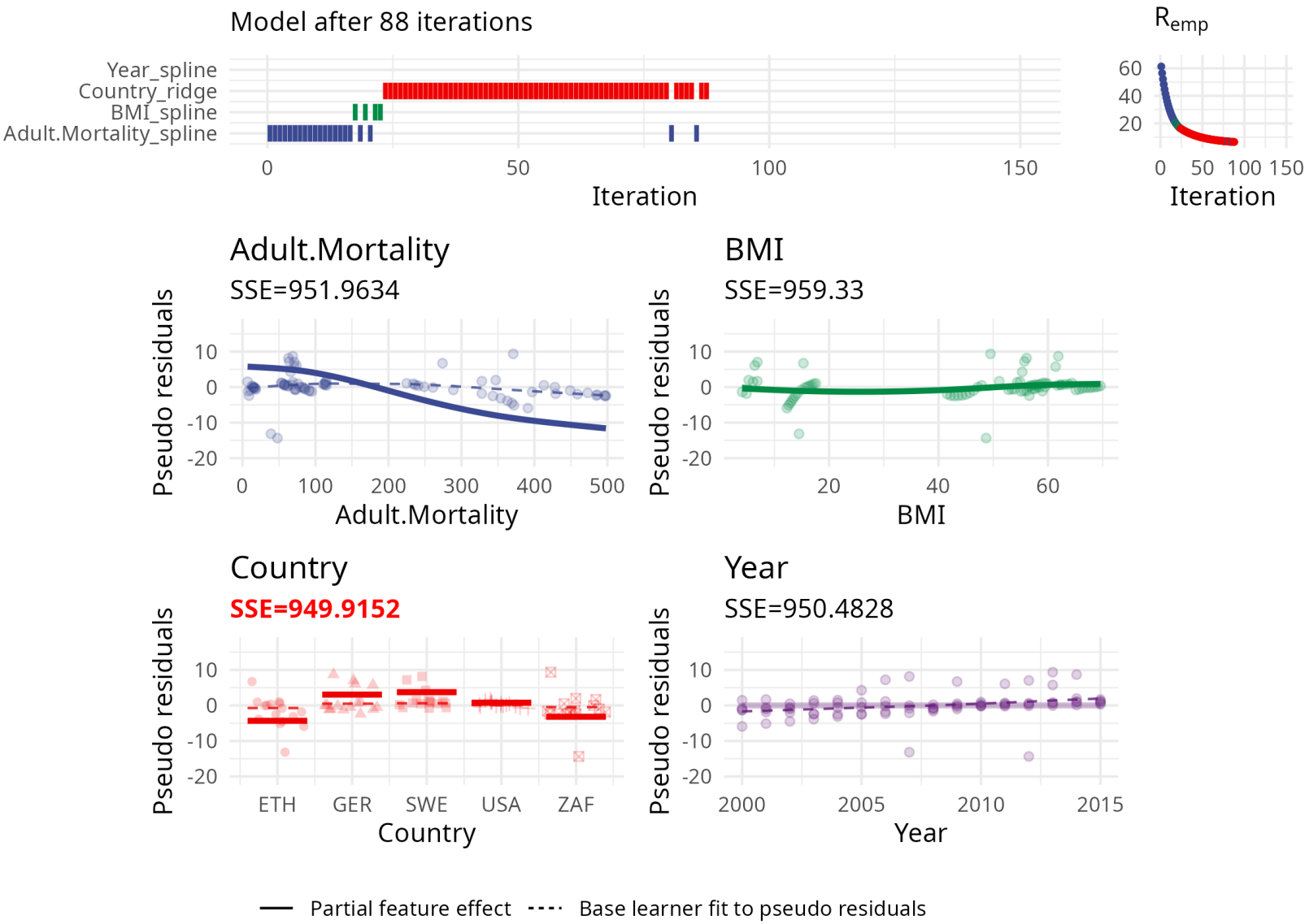
Fitting process: Life expectancy



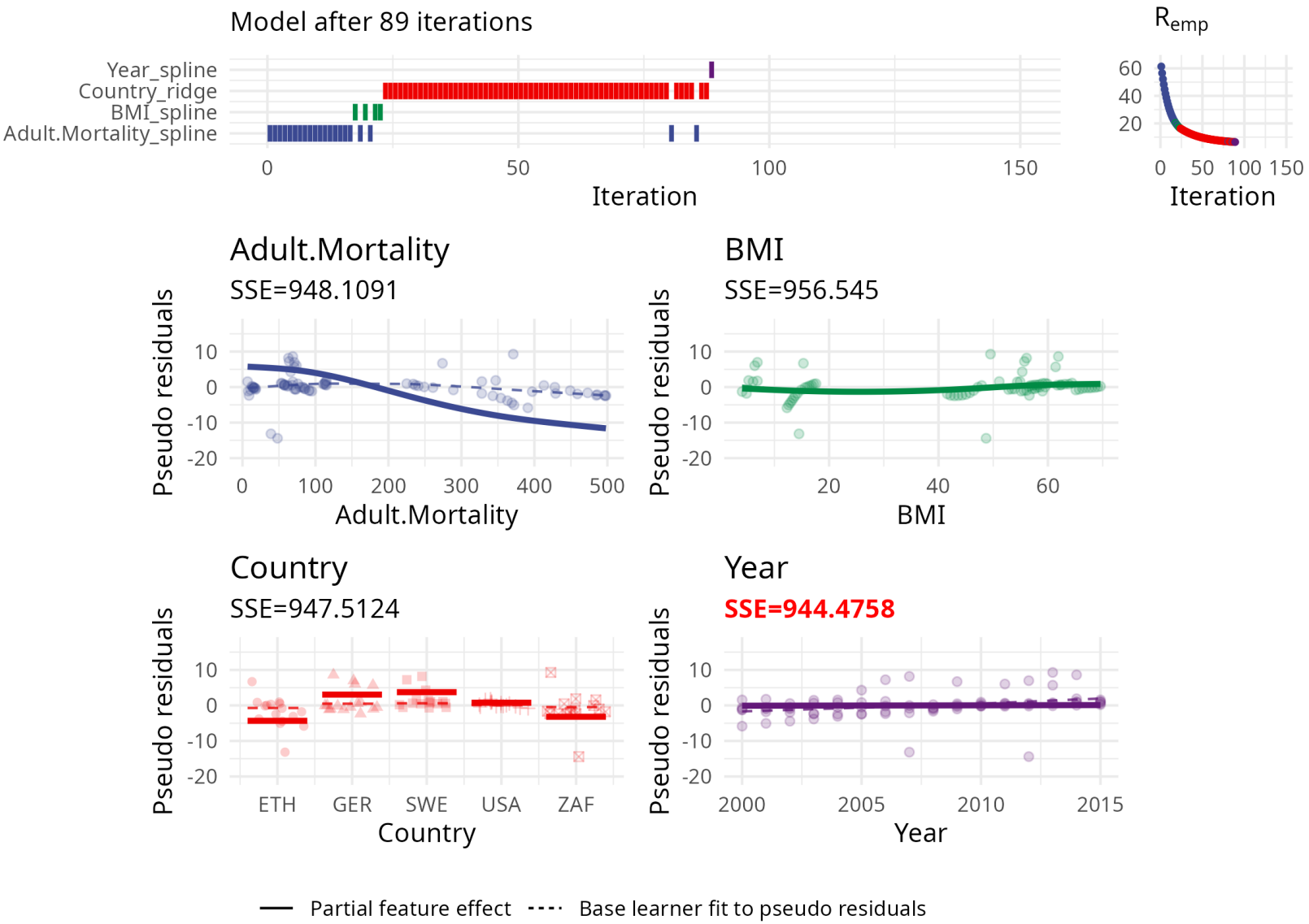
Fitting process: Life expectancy



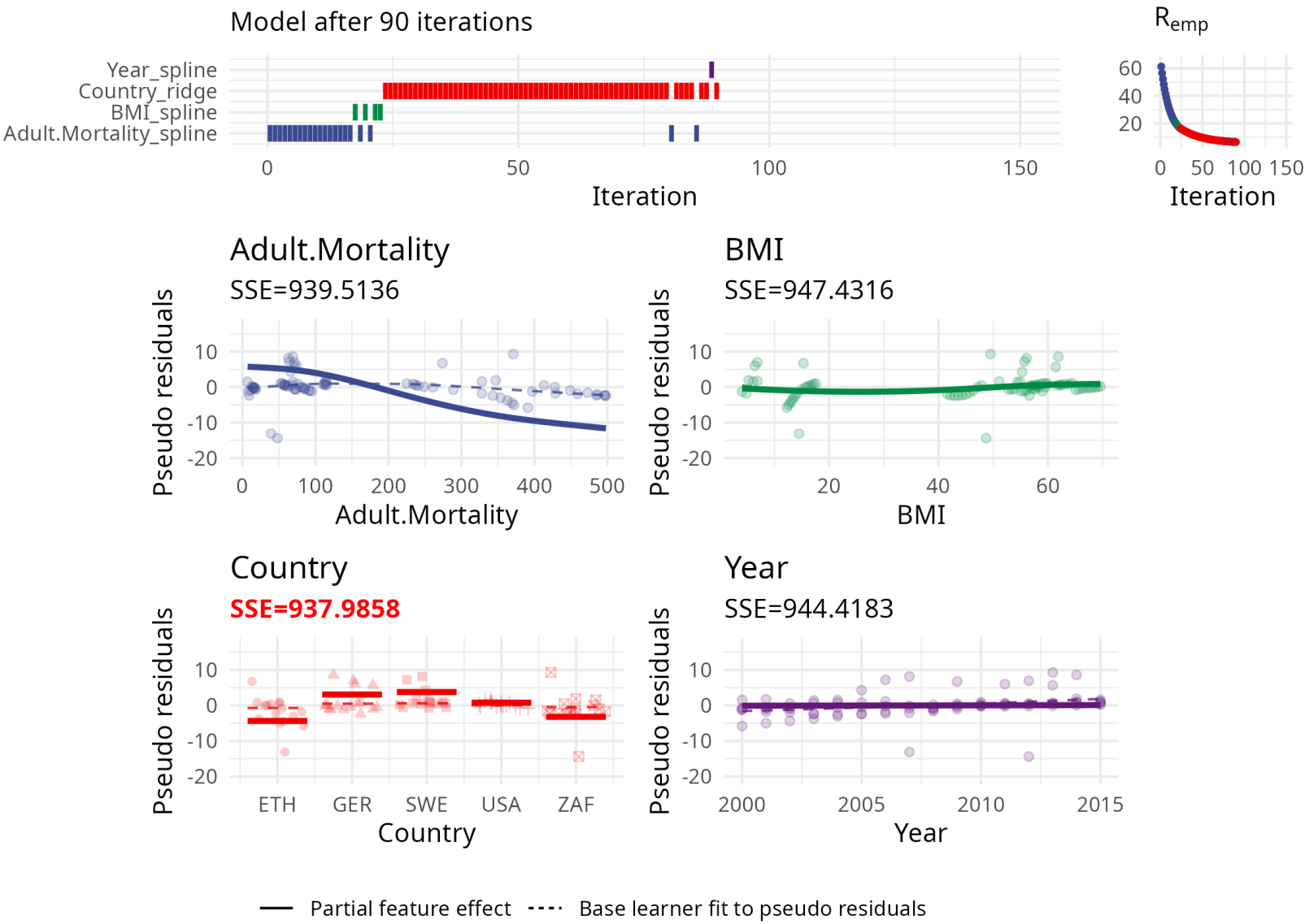
Fitting process: Life expectancy



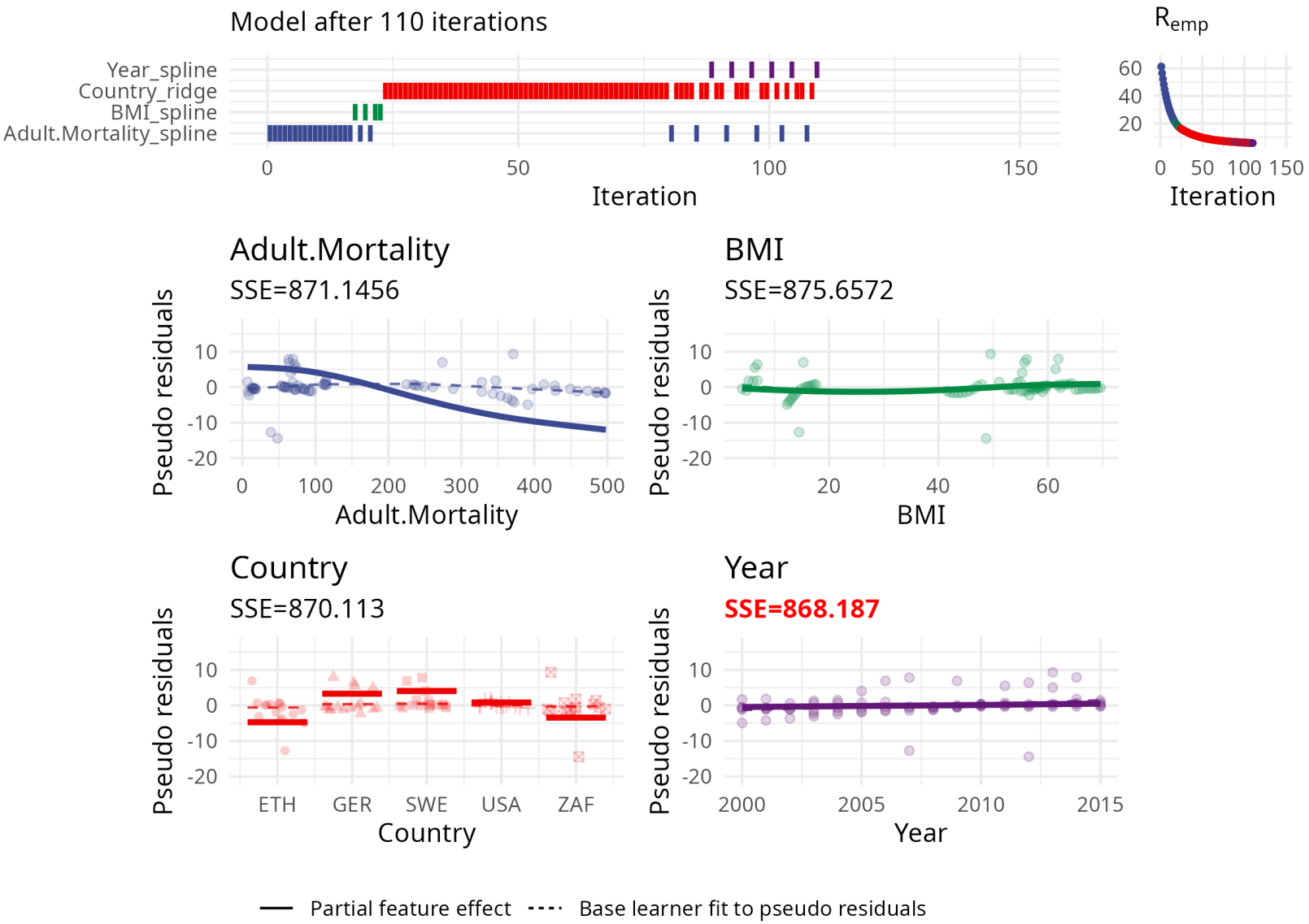
Fitting process: Life expectancy



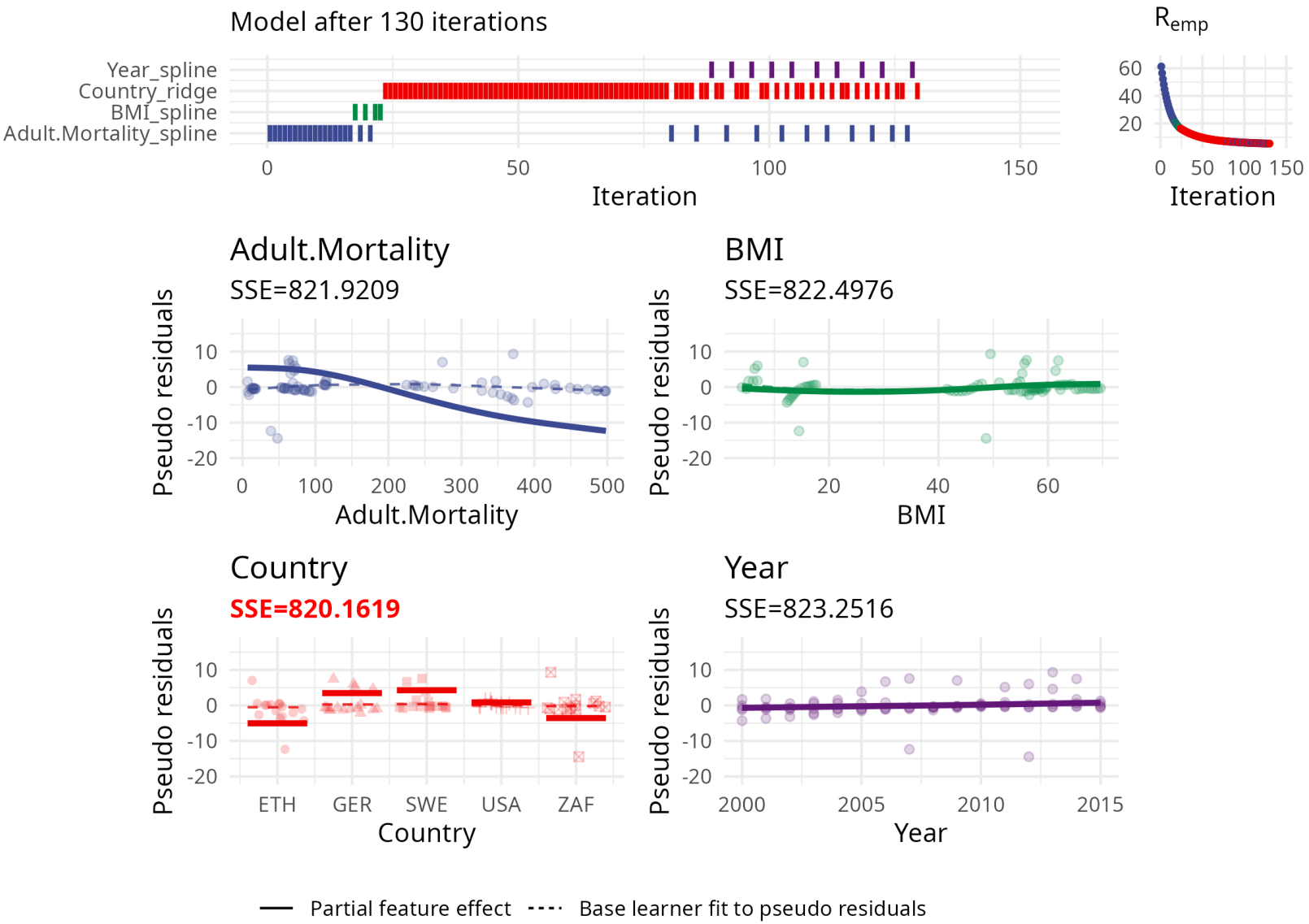
Fitting process: Life expectancy



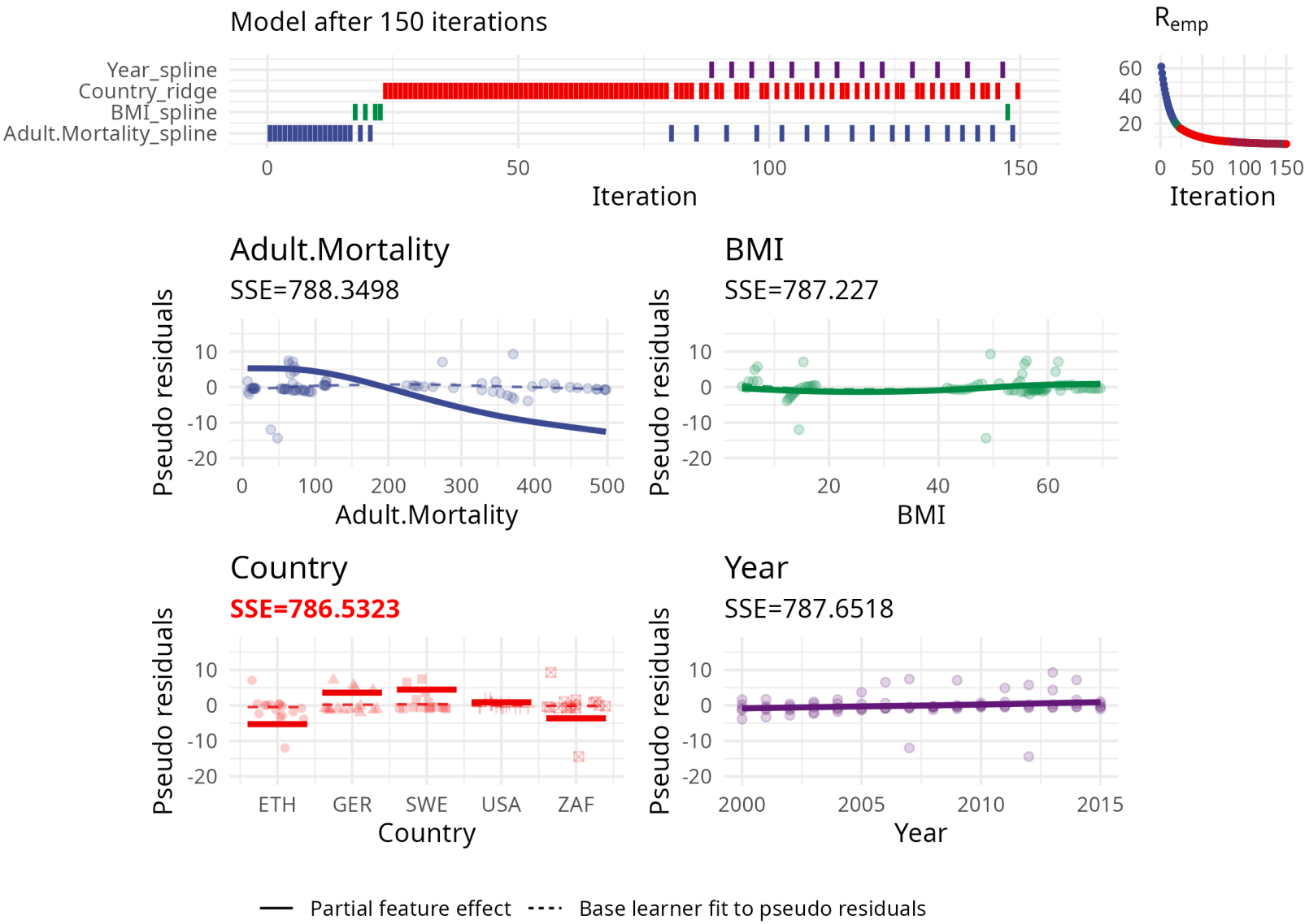
Fitting process: Life expectancy



Fitting process: Life expectancy



Fitting process: Life expectancy



Efficiency

About – Problems of CWB

Computational complexity in terms of memory and runtime efficiency.

- **W.r.t. runtime:**
  - Training CWB requires to (theoretically) calculate $(\boldsymbol{Z}_k^\top \boldsymbol{Z}_k + \boldsymbol{K}_k)^{-1} \boldsymbol{Z}_k^\top \boldsymbol{r}^{[m]}$ for all $k = 1, \ldots, K$ and M iterations.
  - The computational load is tremendously reduced by pre-calculating the Cholesky decompositions of $\boldsymbol{Z}_k^\top \boldsymbol{Z}_k + \boldsymbol{K}_k$ for all k as iteration independent part and to re-cycle these matrices in each iteration.
  - Nevertheless, for big K and M this remains very costly and time consuming.

$\Rightarrow$ Fitting the algorithm can take too much time.

About – Problems of CWB

Computational complexity in terms of memory and runtime efficiency.

- **W.r.t. memory:**
  - Each base learner requires to store a design matrix.
  - For example, using B-splines, it is common to to store an $n \times 24$ matrix (for 20 knots, and cubic basis functions) for one numerical feature.

  $\Rightarrow$ The RAM is filled very fast.

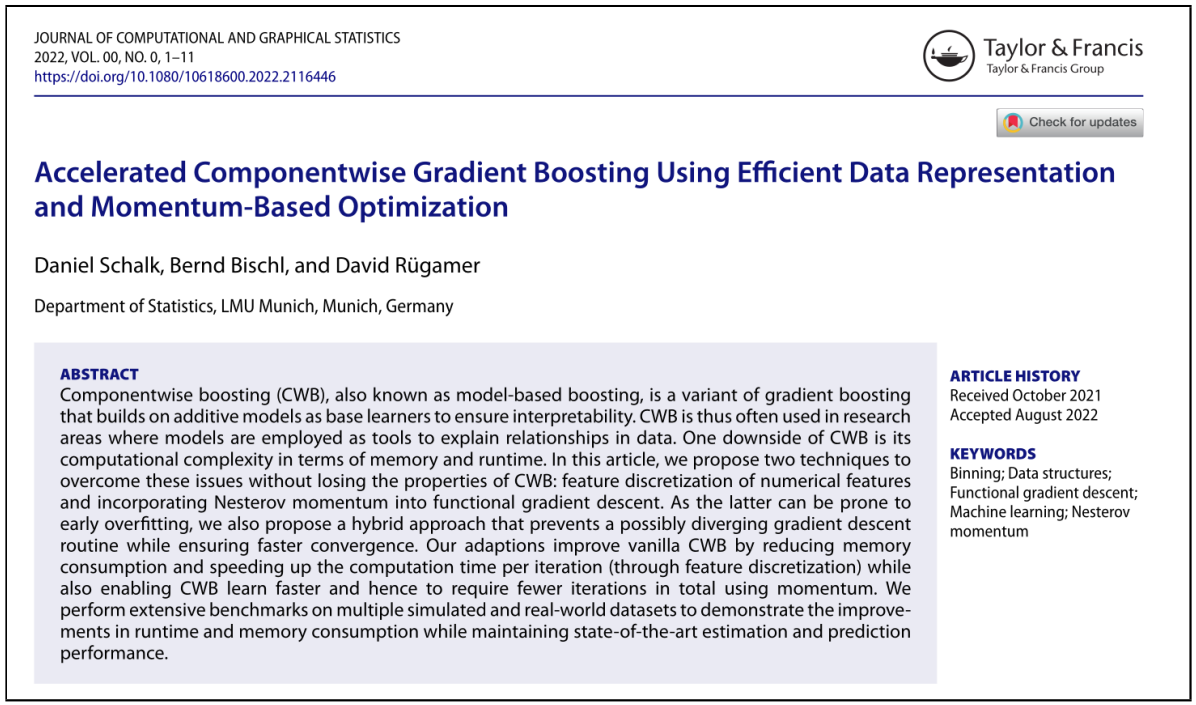
About – Problems of CWB

Computational complexity in terms of memory and runtime efficiency.

- **W.r.t. runtime:** Fitting the algorithm can take too much time.
- **W.r.t. memory:** The RAM is filled very fast.

$\Rightarrow$ Less attractive or infeasible for medium- to large-scale applications.

About – Publication



JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2022, VOL. 00, NO. 0, 1–11
https://doi.org/10.1080/10618600.2022.2116446

Taylor & Francis
Taylor & Francis Group

Check for updates

Accelerated Componentwise Gradient Boosting Using Efficient Data Representation and Momentum-Based Optimization

Daniel Schalk, Bernd Bischl, and David Rügamer

Department of Statistics, LMU Munich, Munich, Germany

ABSTRACT

Componentwise boosting (CWB), also known as model-based boosting, is a variant of gradient boosting that builds on additive models as base learners to ensure interpretability. CWB is thus often used in research areas where models are employed as tools to explain relationships in data. One downside of CWB is its computational complexity in terms of memory and runtime. In this article, we propose two techniques to overcome these issues without losing the properties of CWB: feature discretization of numerical features and incorporating Nesterov momentum into functional gradient descent. As the latter can be prone to early overfitting, we also propose a hybrid approach that prevents a possibly diverging gradient descent routine while ensuring faster convergence. Our adaptions improve vanilla CWB by reducing memory consumption and speeding up the computation time per iteration (through feature discretization) while also enabling CWB learn faster and hence to require fewer iterations in total using momentum. We perform extensive benchmarks on multiple simulated and real-world datasets to demonstrate the improvements in runtime and memory consumption while maintaining state-of-the-art estimation and prediction performance.

ARTICLE HISTORY
Received October 2021
Accepted August 2022

KEYWORDS
Binning; Data structures; Functional gradient descent; Machine learning; Nesterov momentum

Aims:

- **Accelerate the fitting process** of CWB by incorporating Nesterovs momentum.
- **Reduce the memory load** by implementing a more efficient data representation for the design matrices.

Efficiency

Accelerating component-wise boosting

Accelerating component-wise boosting – Idea

Gradient descent:

Parameter space		Function space
$\hat{\boldsymbol{\theta}}^{[m+1]} = \hat{\boldsymbol{\theta}}^{[m]} + \nu \nabla_{\boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\hat{f}(.\|\hat{\boldsymbol{\theta}}^{[m]})\|\mathcal{D})$	$\Rightarrow$	$\hat{f}^{[m+1]} = \hat{f}^{[m]} + \nu \hat{b}^{[m]}$

Nesterovs momentum:

Parameter space		Function space
$\boldsymbol{u}^{[m]} = \nabla_{\boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\hat{f}(.\|\hat{\boldsymbol{\theta}}^{[m]} - \gamma \hat{\boldsymbol{\vartheta}}^{[m-1]})\|\mathcal{D})$ $\hat{\boldsymbol{\vartheta}}^{[m]} = \gamma \hat{\boldsymbol{\vartheta}}^{[m-1]} + \nu \boldsymbol{u}^{[m]}$ $\hat{\boldsymbol{\theta}}^{[m+1]} = \hat{\boldsymbol{\theta}}^{[m]} + \hat{\boldsymbol{\vartheta}}^{[m]}$	$\Rightarrow$	???

$\Rightarrow$ **Idea:** Use Nesterovs momentum and adjust it for functional updates and CWB.

Accelerating component-wise boosting – Idea

- Using momentum and Nesterovs momentum was first proposed by Biau et al. (2019) and refined in an algorithm called Accelerated Gradient Boosting Machine (AGBM) by Lu et al. (2020):

$$
\begin{aligned}
g^{[m]} &= (1 - \theta_m) f^{[m]} + \theta_m h^{[m]} \\
f^{[m+1]} &= g^{[m]} + \eta b^{[m]} \\
h^{[m+1]} &= h^{[m]} + \eta / \theta_m b^{[m]}_{\text{cor}}
\end{aligned}
$$

- Incorporate these adjustments into CWB and make sure all advantages are preserved.

Base learners in AGBM

- $b^{[m]}$ is fit to pseudo residuals $\boldsymbol{r}^{[m]}$ w.r.t. $\hat{g}^{[m-1]}$ instead of $\hat{f}^{[m]}$
- AGBM introduces a second base learner $b_{\text{cor}}^{[m]}$ that is fitted to error-corrected pseudo residuals:

$$c^{[m](i)} = r^{[m](i)} + \frac{m}{m+1}(c^{[m-1](i)} - \hat{b}_{\text{cor}}^{[m-1]}(\mathbf{x}^{(i)})),$$

with $i = 1, \ldots, n$, if $m > 1$ and $\boldsymbol{c}^{[m]} = \boldsymbol{r}^{[m]}$ if $m = 0$.

$\Rightarrow$ Each iteration adds not one but two base learners $b^{[m]}$ and $b_{\text{cor}}^{[m]}$:

  - $b_{\text{cor}}^{[m]}$ defines the momentum sequence to accelerate the fitting into the direction of the error-corrected pseudo residuals
  - Computing a second base learner also means two double the runtime for the same number of iterations.

Accelerating component-wise boosting

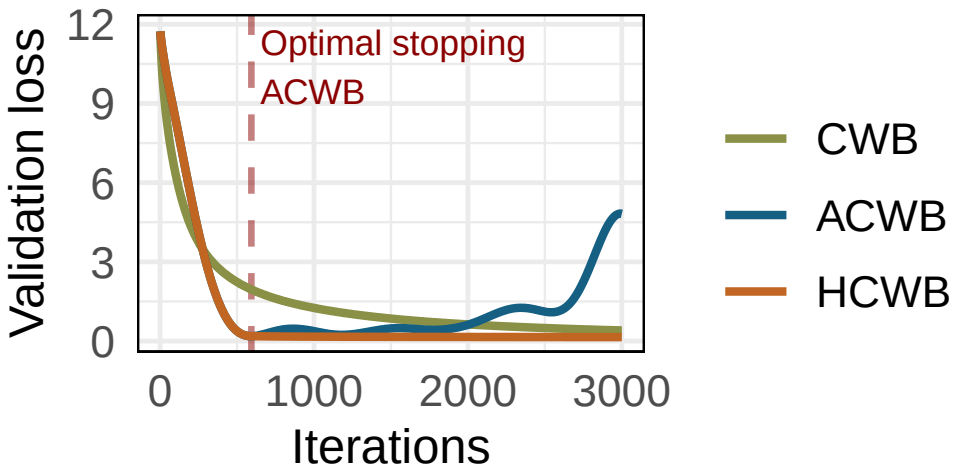
Schalk et al. (2022a) introduces an accelerated CWB (ACWB) version by incorporating these adaptions to CWB, therefore:

- Both base learners, $b^{[m]}$ and $b_{\text{cor}}^{[m]}$, are the result of a selection process that chooses one of $b_1, \ldots, b_K$ w.r.t. to the minimal SSE on the respective pseudo residuals $\boldsymbol{r}^{[m]}$ and $\boldsymbol{c}^{[m]}$.
- Update the estimated parameters accordingly to allow the estimation of partial feature effects.

Considering these points allows to maintain all advantages of CWB in ACWB. Details are outlined in the publication.

Hybrid component-wise boosting

- It was proven by Lu et al. (2020), that ACWB can overfit if not stopped early.
- Therefore, a hybrid CWB (HCWB) approach combines ACWB for an accelerated fitting in the beginning and CWB to fine-tune the model:



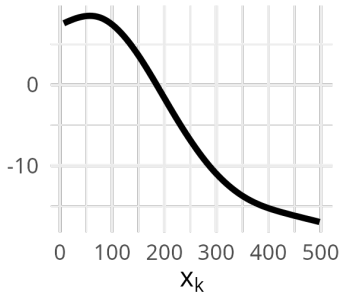
Efficiency

Reduced memory consumption in component-wise boosting

Base learner design matrix

- Each base learner $b_1, \ldots, b_K$ requires to build a design matrix $\boldsymbol{Z}_k \in \mathbb{R}^{n \times d_k}$ based on the feature vector $\boldsymbol{x}_k$.
- For example:

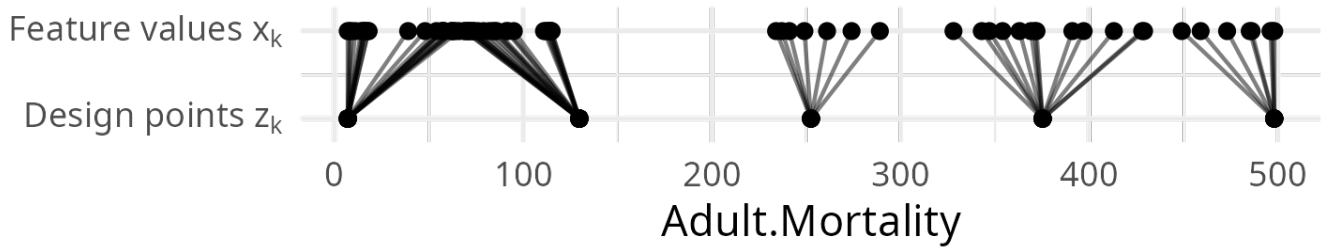
$$\underbrace{\begin{pmatrix} 234 \\ 73 \\ 498 \\ \vdots \\ 112 \\ 261 \\ 343 \end{pmatrix}}_{=\boldsymbol{x}_k \in \mathbb{R}^n} \Rightarrow \underbrace{\begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.07 & 0.62 & 0.31 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.20 & 0.66 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 & 0.40 & 0.55 & 0.04 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.63 & 0.08 & 0.00 & 0.00 \end{pmatrix}}_{=\boldsymbol{Z}_k \in \mathbb{R}^{n \times d_k} \text{ (B-spline basis)}} \Rightarrow b_k$$



$\Rightarrow$ If n is large, the memory gets filled very fast (especially if p is also large).

Binning

- To reduce the memory consumption, we applied binning to operate on a reduced representation of Z_k.
- Binning is a technique that allows to represent the n values $x_k^{(1)}, \ldots, x_k^{(n)}$ of $\mathbf{x}_k$ by $n^* < n$ design points $\boldsymbol{z}_k = (z_k^{(1)}, \ldots, z_k^{(n^*)})$.
- The idea is to assign each $x_k^{(i)}$ to the closest design point $z_k^{(i)}$ and store the assignment in a map $\text{ind}_k^{(i)}$: $x_k^{(i)} \approx z_k^{(\text{ind}_k^{(i)})}$



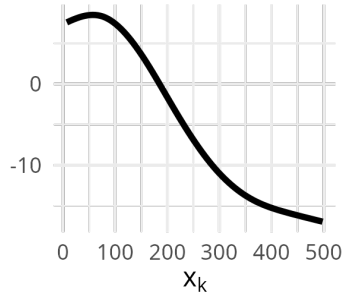
Binning in GLMs

- Lang et al. (2014) used binning to discretize feature vectors to increase the efficiency of multilevel structured additive regression.
- Wood et al. (2017) applied binning to fit GAMs to gigadata and argue that the best approximation is achieved by setting $n^* = \sqrt{n}$.
- Li and Wood (2020) presented optimized cross-product operations of binned design matrices to also speed up the fitting.

Binning in CWB

- Represent numerical features $\boldsymbol{x}_k$ by n^* design points $\boldsymbol{z}_k$.
- Build the design matrix $\boldsymbol{Z}_k^*$ based on $\boldsymbol{z}_k$ which requires to store $n^* d_k$ values instead of $n d_k$.
- Use optimized cross-product operations to estimate the parameters $\hat{\boldsymbol{\theta}}_k^{[m]}$ of base learner b_k to also speed up the fitting.
- TODO: Hier sollte noch irgendwo dazu, dass das für jeden base learner der ein numerisches Feature handelt individuell gemacht wird.

$$\underbrace{\begin{pmatrix} 7.0 \\ 129.8 \\ 252.5 \\ 375.2 \\ 498.0 \end{pmatrix}}_{=\boldsymbol{z}_k \in \mathbb{R}^{n^*}} \Rightarrow \underbrace{\begin{pmatrix} 0.17 & 0.67 & 0.17 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.32 & 0.61 & 0.07 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.02 & 0.48 & 0.48 & 0.02 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.07 & 0.61 & 0.32 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.67 & 0.17 \end{pmatrix}}_{=\boldsymbol{Z}_k^* \in \mathbb{R}^{n^* \times d_k} \text{ (B-spline basis)}} \Rightarrow b_k$$



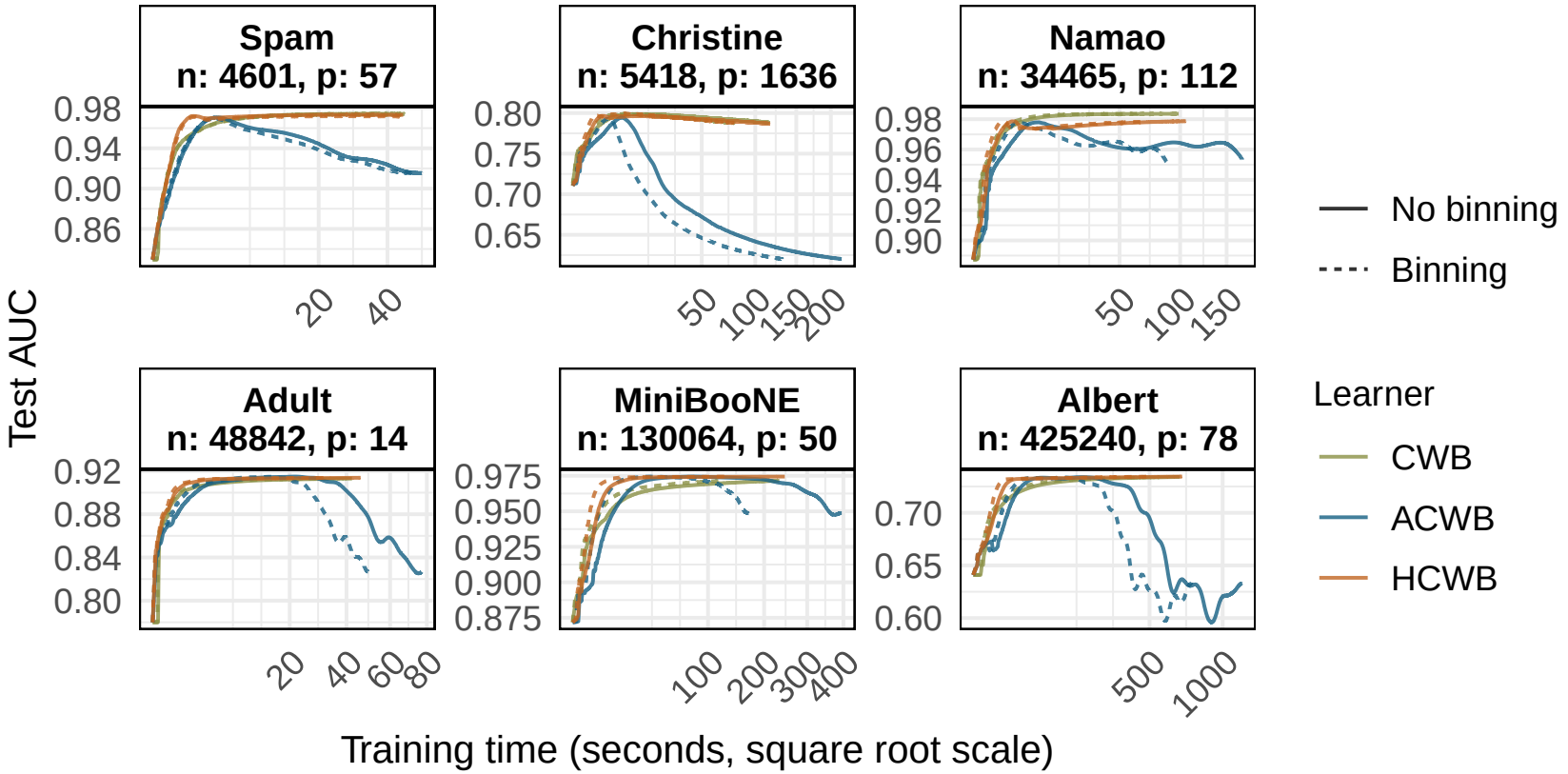
Example: Memory savings

Tabelle wie viel Speicher wann gespart wird.

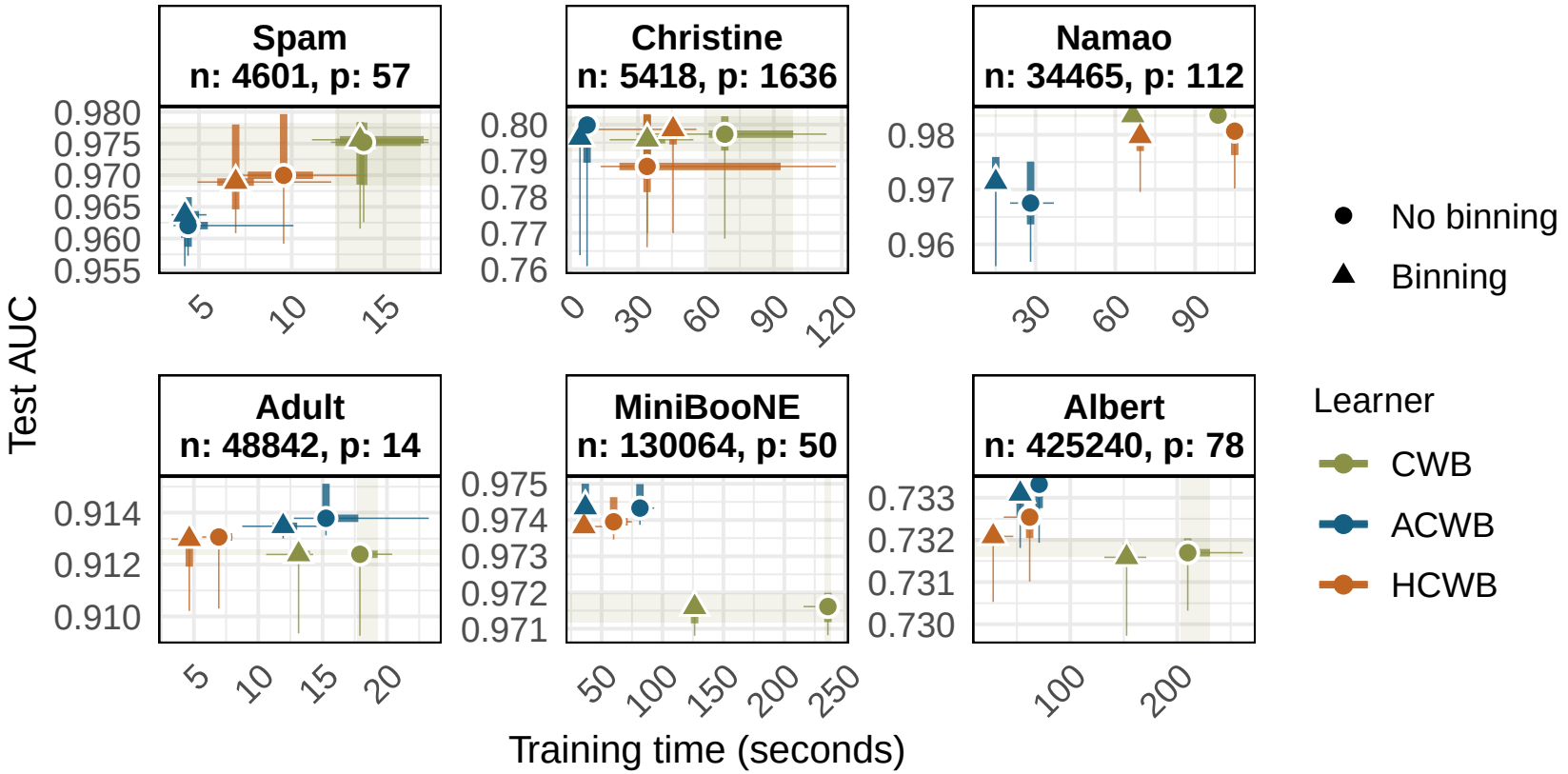
Efficiency

Results and Summary

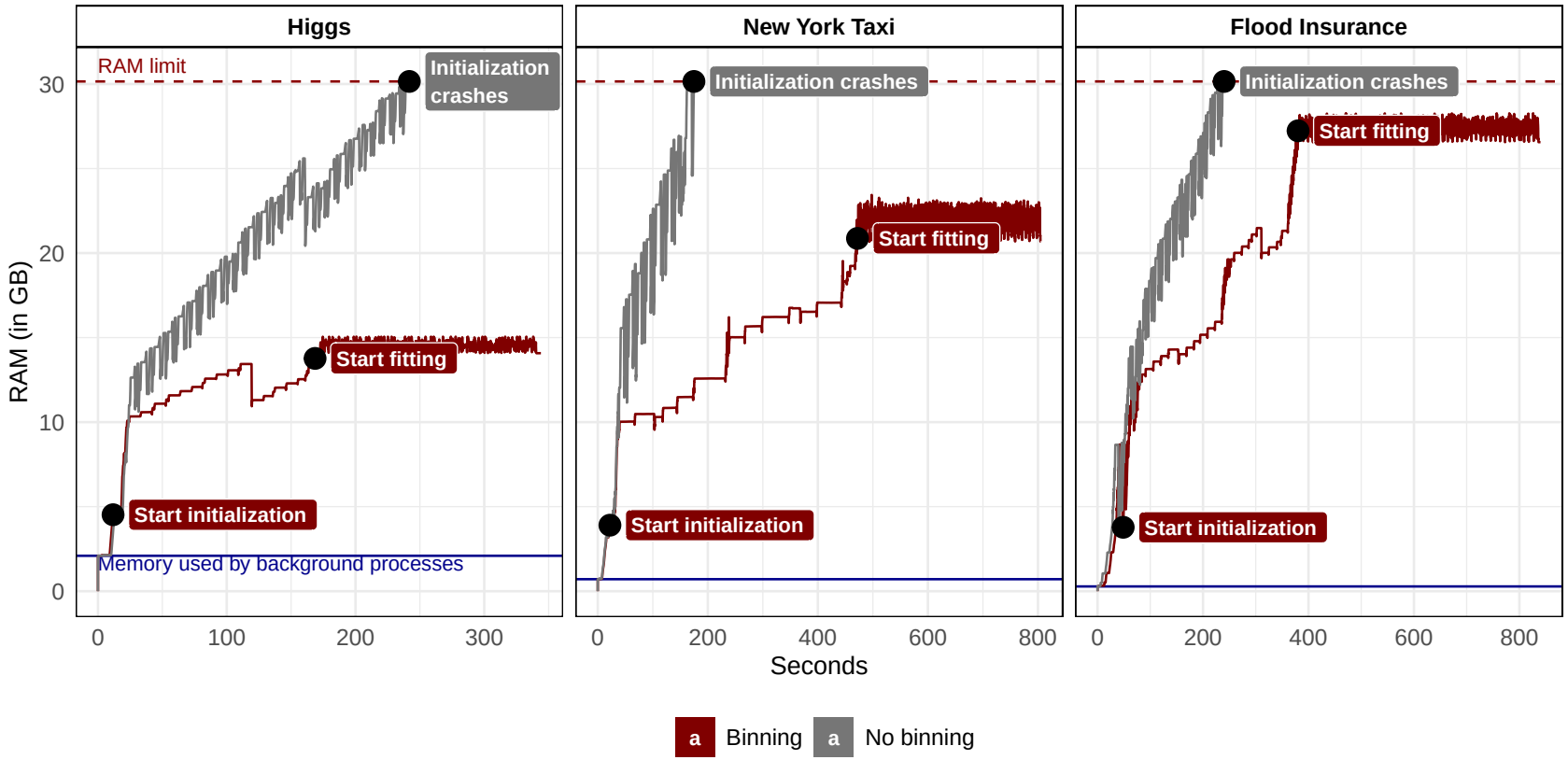
Results – Runtime comparisons of CWB variants



Results – Benchmark comparisons of CWB variants



Results – Memory consumption for bigger data sets



Summary

- We incorporated Nesterovs momentum to speed up the fitting time.
- Binning allows to save memory by a reduced representation and also to speed up the fitting process by using tailored matrix operations.
- In a simulation study, we tried to find good default values for the momentum for both (ACWB: 0.0034 and HCWB: 0.037) algorithms.
- A benchmark showed the effectiveness of these approaches by being significantly faster by achieving the same or better test performance.
- The adaptions are implemented in `compboost`.

Outlook

- Using array arithmetic to even faster calculate matrix products and tensor product base learner.

Distributed computing

Distrirbuted data set

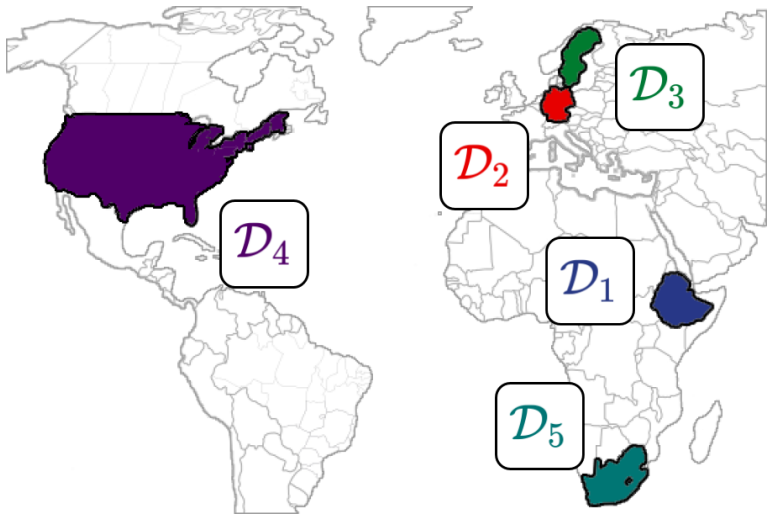
Assume the country column is not present in the data set and each country holds a its own partition:

Life.expectancy	Country	Year	BMI	Adult.Mortality	Data set
51.2	ETH	2000	12.3	391	$\mathcal{D}_1$
⋮	⋮	⋮	⋮	⋮	
64.8	ETH	2015	17.6	225	
78.0	GER	2000	55.1	95	$\mathcal{D}_2$
⋮	⋮	⋮	⋮	⋮	
81.0	GER	2015	62.3	68	
79.6	SWE	2000	52.8	73	$\mathcal{D}_3$
⋮	⋮	⋮	⋮	⋮	
82.4	SWE	2015	59.5	53	
76.8	USA	2000	6.1	114	$\mathcal{D}_4$
⋮	⋮	⋮	⋮	⋮	
79.3	USA	2015	69.6	13	
57.3	ZAF	2000	4.1	397	$\mathcal{D}_5$
⋮	⋮	⋮	⋮	⋮	
62.9	ZAF	2015	51.1	328	

$\Rightarrow$ The data set is distributed over S (here $S = 5$) sites.

Distrirbuted data set

Assume the country column is not present in the data set and each country holds a its own partition:



- We are looking at **horizontally partitioned data**: Each site holds the same features but different observations.
- **Vertically partitioned data**: Each site has the same observations but different features.

$\Rightarrow$ Can we still fit a model with CWB?

Distributed setup

- S sites, each exclusively hold a data set $\mathcal{D}_s$
- A host in the middle controls the communication with the sites, the sites cannot communicate with each other.
- The host is the volnurable component since it can be highjacket and access to the communicated data is ont out of question.
- Hence, the communicated data must ensure privacy of the original data sets.

Non-disclosive data

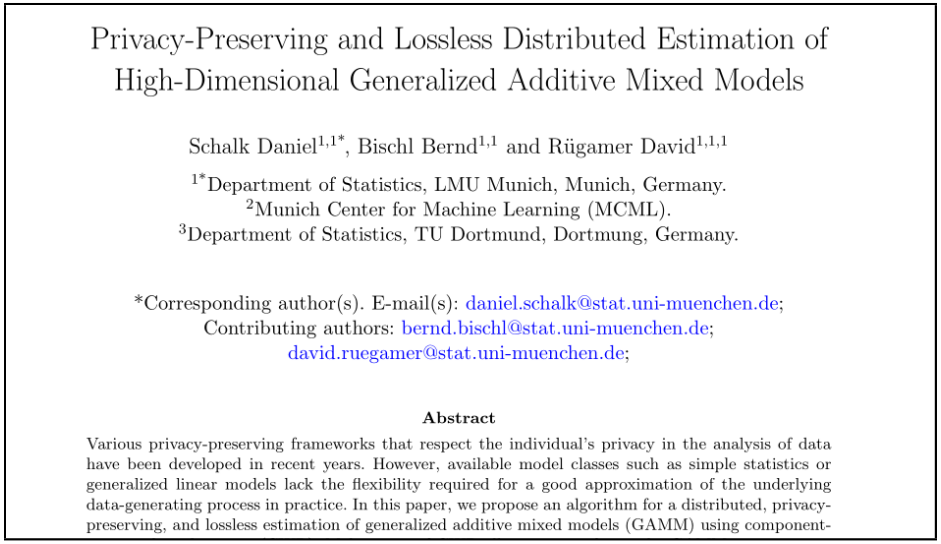
- The entire data set or parts of it are not allowed to get shared between sites or with third parties (like analysts).
- Often applies to sensitive data, e.g., most data sets with private information about individuals.

$\Rightarrow$ It is not possible to merge the data sets $\mathcal{D} = \cup_{s=1}^{S} \mathcal{D}_s$ at a global server/machine/location.

What is allowed to get shared?

- Aggregated data that does not allow to re-construct parts of the original data set.
- Encrypted data (e.g. via homomorphic encription (Gentry, 2009)).

About – Publication



Privacy-Preserving and Lossless Distributed Estimation of High-Dimensional Generalized Additive Mixed Models

Schalk Daniel[1,1*], Bischl Bernd[1,1] and Rügamer David[1,1,1]

[1*]Department of Statistics, LMU Munich, Munich, Germany.
[2]Munich Center for Machine Learning (MCML).
[3]Department of Statistics, TU Dortmund, Dortmung, Germany.

*Corresponding author(s). E-mail(s): daniel.schalk@stat.uni-muenchen.de;
Contributing authors: bernd.bischl@stat.uni-muenchen.de;
david.ruegamer@stat.uni-muenchen.de;

Abstract

Various privacy-preserving frameworks that respect the individual's privacy in the analysis of data have been developed in recent years. However, available model classes such as simple statistics or generalized linear models lack the flexibility required for a good approximation of the underlying data-generating process in practice. In this paper, we propose an algorithm for a distributed, privacy-preserving, and lossless estimation of generalized additive mixed models (GAMM) using component-

Aims:

- Provide a **distributed and lossless CWB algorithm**:

$$\text{distCWB}(\mathcal{D}_1, \ldots, \mathcal{D}_S, \ldots) = \text{CWB}(\mathcal{D}, \ldots)$$

- **Allow for site-specific corrections** get a deeper understanding of the distributed data.

Site-specific vs. main effects

- Reminder: CWB can be used to fit a GAM $f(\mathbf{x}) = f_0 + \sum_{k=1}^{K} b_k(\mathbf{x})$ with the base learners b_k as additive terms.
- In the distributed setup, we denote b_k as shared or main effect that is equal between all sites.
- Furhter, these effects are extended by site-specific effects $b_{k,s}$ that allow a site-specific correction of the shared effect.

The final model assembles both, common shared and site specific effects:

$$f(\mathbf{x}) = f_0 + \sum_{k=1}^{K} \left(b_k(\mathbf{x}) + \sum_{s=1}^{S} b_{k,s}(\mathbf{x}) \right) = f_0 + \sum_{k=1}^{K} b_k(\mathbf{x}) + (b_k \odot b_0)(\mathbf{x})$$

This structure is equal to CWB with base learner b_k, $k = 1, \ldots, K$, and tensor product base learners $b_k \odot b_0$, $k = 1, \ldots, K$, with b_0 a latent one hot encoded categorical base learner modelling the S sites.

Distributed base learner estimation

We restricted CWB to fit b_k using OLS by $\hat{\boldsymbol{\theta}}_k = (\boldsymbol{Z}_k^\top \boldsymbol{Z}_k)^{-1} \boldsymbol{Z}_k^\top \boldsymbol{r}^{[m]}$.

Algorithm 3 Vanilla CWB algorithm

Input Train data $\mathcal{D}$, learning rate ν, number of boosting iterations M, loss function L, base learners $b_1, \ldots, b_K$

Output Model $\hat{f} = \hat{f}^{[M]}$

procedure CWB($\mathcal{D}, \nu, M, L, b_1, \ldots, b_K$)

Initialize: $f_0 = \hat{f}^{[0]}(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} \mathcal{R}_{\text{emp}}(c|\mathcal{D})$

while $m \leq M$ **do**

$$r^{[m](i)} = - \left. \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})} \right|_{f=\hat{f}^{[m-1]}}, \quad \forall i \in \{1, \ldots, n\}$$

for $k \in \{1, \ldots, K\}$ **do**

$$\hat{\boldsymbol{\theta}}_k^{[m]} = \left(\boldsymbol{Z}_k^\top \boldsymbol{Z}_k + \boldsymbol{K}_k \right)^{-1} \boldsymbol{Z}_k^\top \boldsymbol{r}^{[m]} \quad \leftarrow$$

$\text{SSE}_k = \sum_{i=1}^n (r^{[m](i)} - b_k(\mathbf{x}^{(i)}|\hat{\boldsymbol{\theta}}_k^{[m]}))^2$

$k^{[m]} = \arg\min_{k \in \{1, \ldots, K\}} \text{SSE}_k$

$\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \nu b_{k^{[m]}}(\mathbf{x}|\hat{\boldsymbol{\theta}}_{k^{[m]}}^{[m]})$

return $\hat{f} = \hat{f}^{[M]}$

$\Rightarrow$ Fitting a linear model to horizontally distributed data allows to

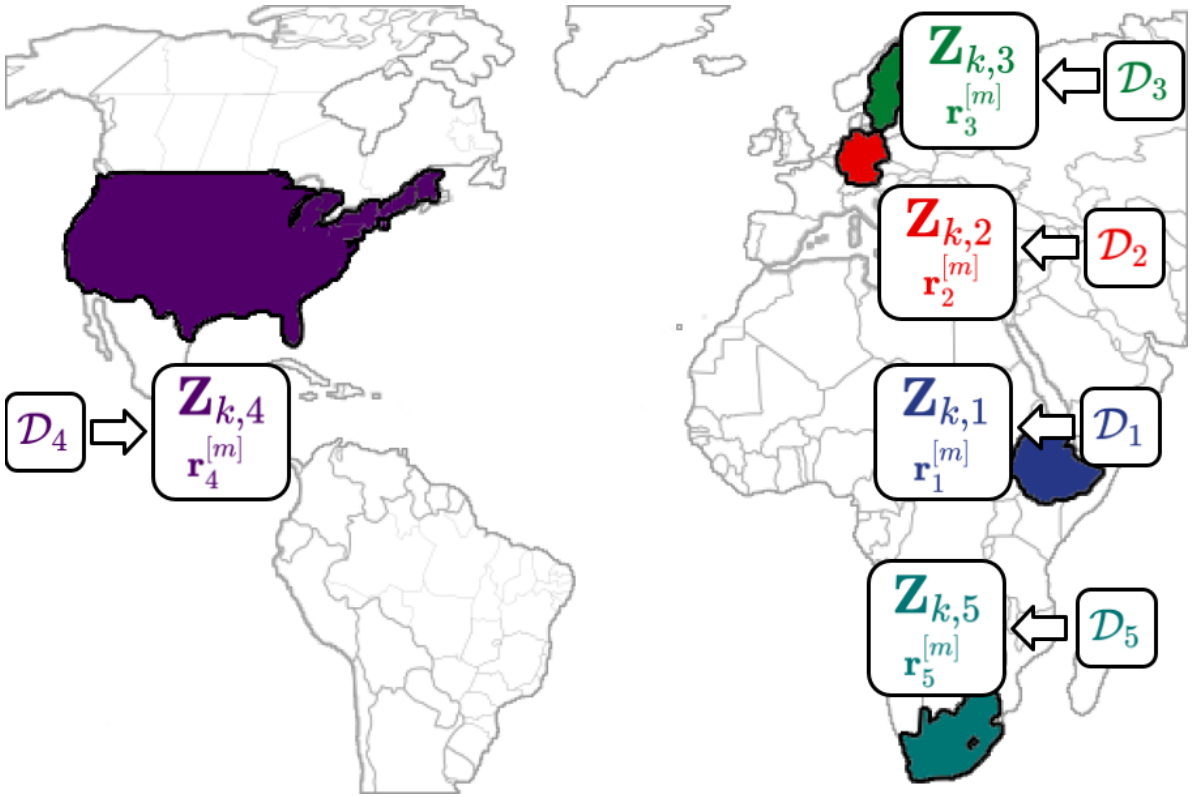
Distributed linear model

We restricted CWB to fit b_k as a linear model in each iteration m as $\hat{\boldsymbol{\theta}}_k = (\boldsymbol{Z}_k^\top \boldsymbol{Z}_k)^{-1} \boldsymbol{Z}_k^\top \boldsymbol{r}^{[m]}$.

$\Rightarrow$ Fitting a linear model to horizontally distributed data allows to also fit the base learners.

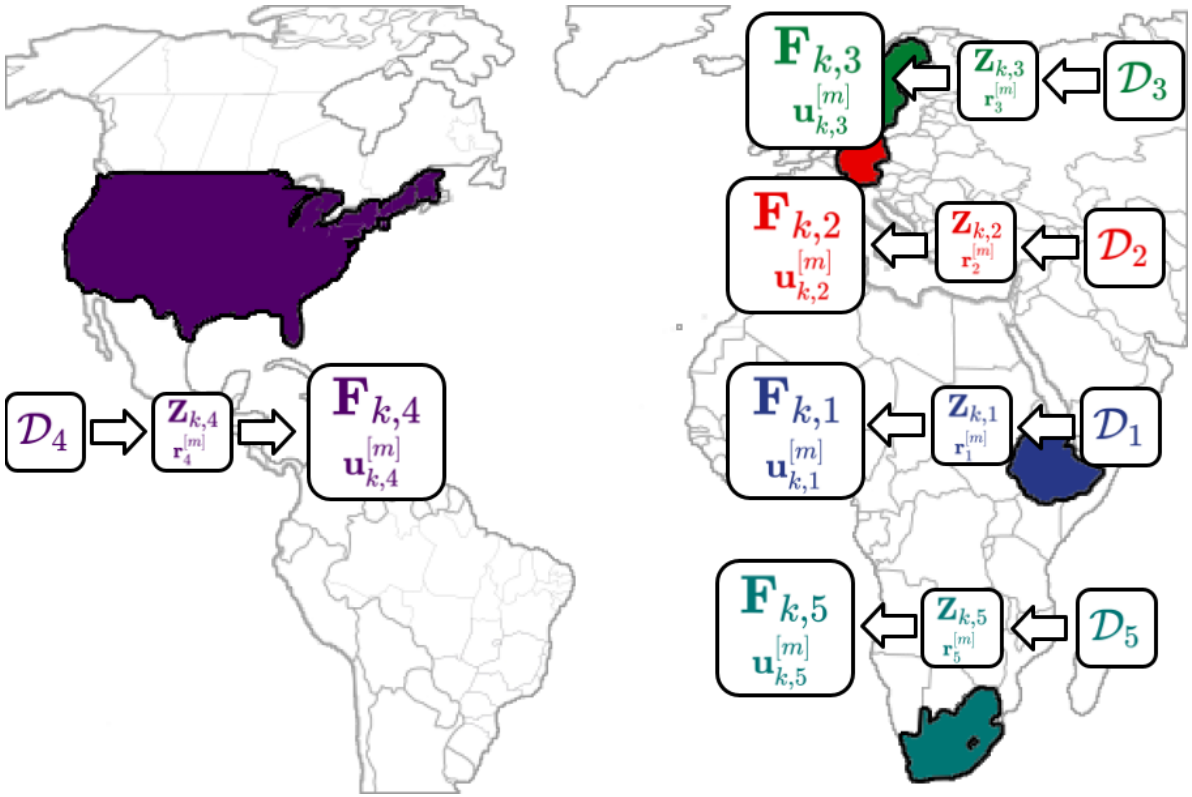
Distributed linear model

- Each site holds a design matrix for the k base learner: $Z_{k,s}$ and a slice of the pseudo residuals: $r_s^{[m]}$



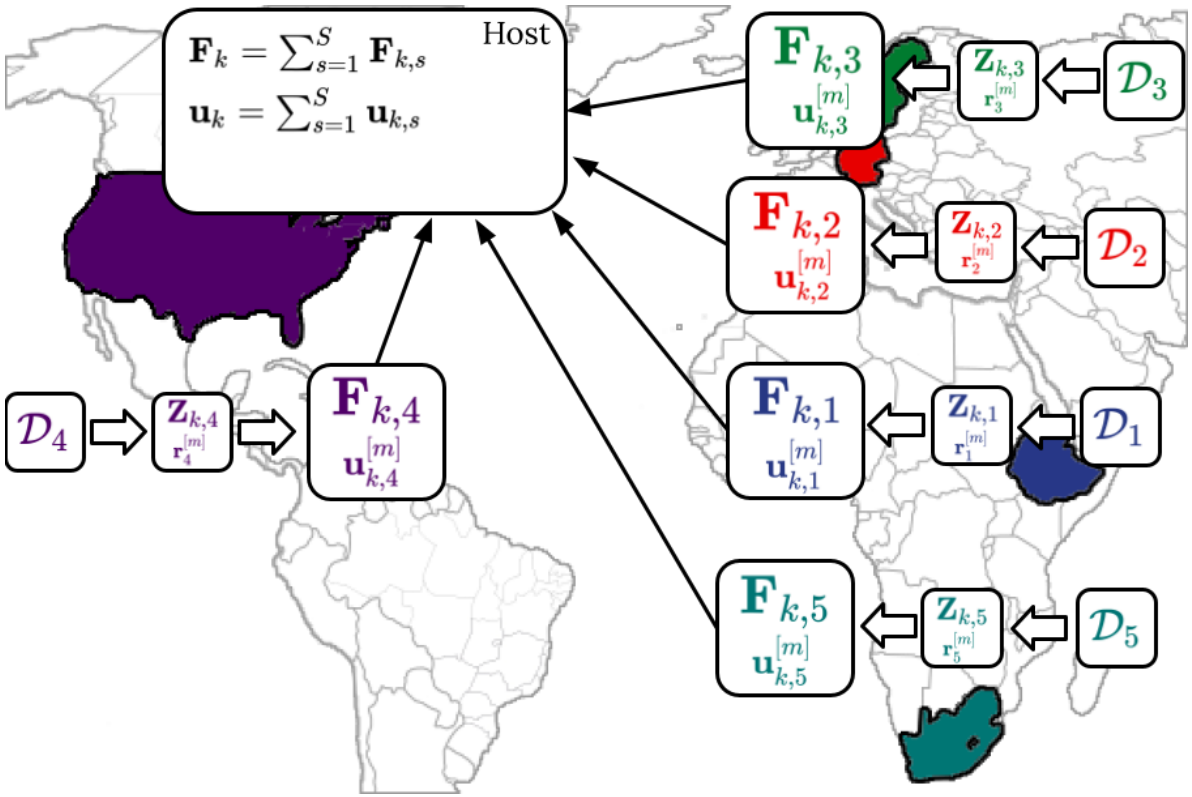
Distributed linear model

- Each site calculates $F_{k,s} = Z_{k,s}^T Z_{k,s}$ and $u_{k,s}^{[m]} = Z_{k,s}^T r_k^{[m]}$.



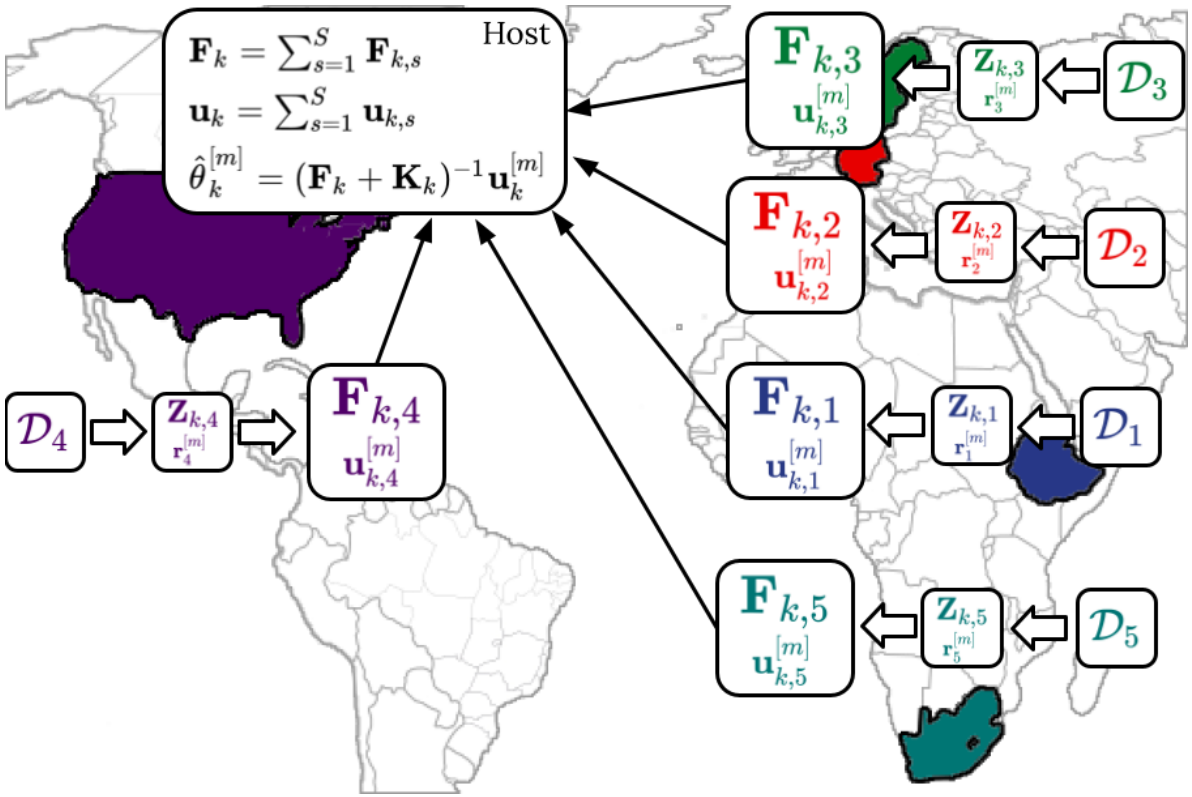
Distributed linear model

- The sites are allowed to communicate $F_{k,s}$ and $u_{k,s}^{[m]}$ as long as "enough observations" are used.



Distributed linear model

- That allows to gather $F_{k,s}$, and $u_{k,s}^{[m]}$, $s = 1, \ldots, S$, calculate $F_k = \sum_{s=1}^{S} F_{k,s}$ and $u_k^{[m]} = \sum_{s=1}^{S} u_{k,s}$, and to estimate $\hat{\theta}_k^{[m]} = (F_k + K_k)^{-1} u_k$.



Algorithm 4 Distributed Effect Estimation (Karr et al., 2005).
The line prefixes `[S]` and `[H]` indicate whether the operation is conducted at the sites (`[S]`) or at the host (`[H]`).

Input Sites design matrices $\boldsymbol{Z}_{k,1}, \ldots, \boldsymbol{Z}_{k,S}$, response vectors $\boldsymbol{r}_1^{[m]}, \ldots, \boldsymbol{r}_S^{[m]}$ and an optional penalty matrix $\boldsymbol{K}_k$.
Output Estimated parameter vector $_k$.

procedure distFit($\boldsymbol{Z}_{k,1}, \ldots, \boldsymbol{Z}_{k,S}, \boldsymbol{r}_1^{[m]}, \ldots, \boldsymbol{r}_S^{[m]}, \boldsymbol{K}_k$)
  for $s \in \{1, \ldots, S\}$ **do**
    `[S]` $\boldsymbol{F}_{k,s} = \boldsymbol{Z}_{k,s}^\top \boldsymbol{Z}_{k,s}$
    `[S]` $\boldsymbol{u}_{k,s} = \boldsymbol{Z}_{k,s}^\top \boldsymbol{r}_s^{[m]}$
    `[S]` Communicate $\boldsymbol{F}_{k,s}$ and $\boldsymbol{u}_{k,s}$ to the host
  `[H]` $\boldsymbol{F}_k = \sum_{s=1}^{S} \boldsymbol{F}_{k,s} + \boldsymbol{K}_k$
  `[H]` $\boldsymbol{u}_k = \sum_{s=1}^{S} \boldsymbol{u}_{k,s}$
  `[H]` **return** $_k = \boldsymbol{F}_k^{-1} \boldsymbol{u}_k$

Distributed fitting of main effects in CWB

- In each iteration m, the main effect is estimated by $\hat{\boldsymbol{\theta}}_k^{[m]} = \text{distFit}(Z_{k,1}, \ldots, Z_{k,S}, \mathbf{y}_1, \ldots, \mathbf{y}_S, K_k) bla$

Distributed fitting of site-specific effects

- Site-specific effects sind etwas ”einfacher” als base learner per site zu schätzen
- parametercshätzer teilen
- Zeigen, dass der tensor spline learner einfach als einzelne Fits ausgedrückt werden kann.

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{k_\times} &= \left(Z_{k_\times}^\top Z_{k_\times} + K_{k_\times}\right)^{-1} Z_{k_\times}^\top \mathbf{y} \\ &= \begin{pmatrix} (Z_{k,1}^\top Z_{k,1} + \lambda_0 I_{d_k} + K_k)^{-1} Z_{k,1}^\top \mathbf{y}_1 \\ \vdots \\ (Z_{k,S}^\top Z_{k,S} + \lambda_0 I_{d_k} + K_k)^{-1} Z_{l,S}^\top \mathbf{y}_S \end{pmatrix}\end{aligned}$$

$$\hat{\boldsymbol{\theta}}_{k_\times,s} = (Z_{k,s}^\top Z_{k,s} + \lambda_0 I_{d_k} + K_k)^{-1} Z_{k,s}^\top \mathbf{y}_s$$

Distributed component-wise gradient boosting

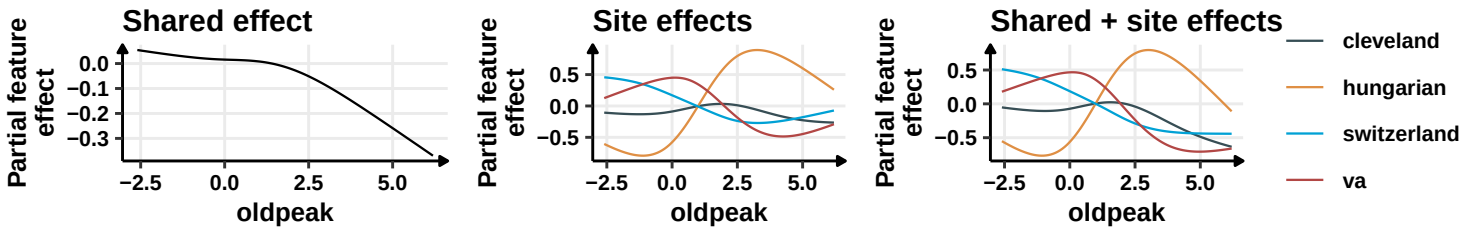
- How is everything put together?
- Algorithmus zeigen, vielleicht eine iteration mit dem ganzen teilen etc. in die Karte einzeichnen?
- Kurz communication costs zeigen?
- Implementiert mit DataSHIELD.

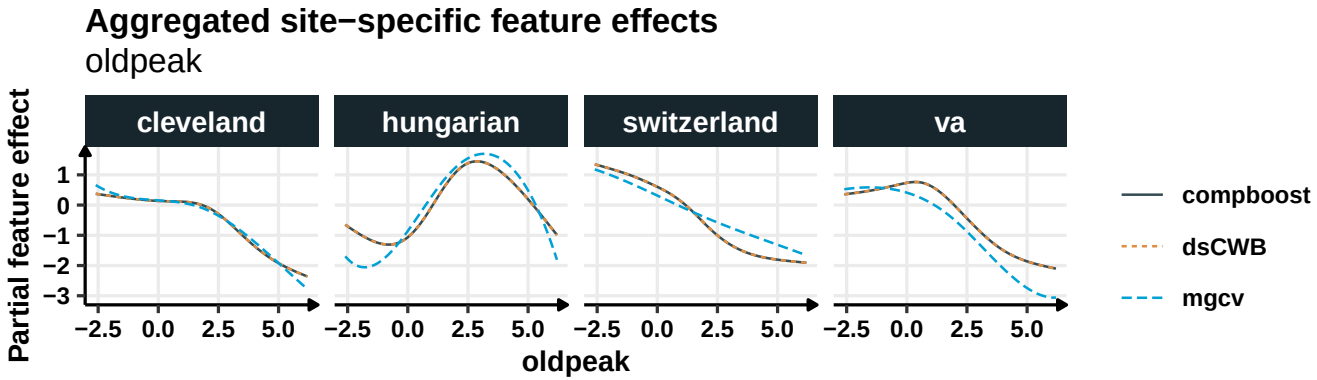
Efficiency

Comparisons, summary, and outlook

Comparisons

- **Reminder:** The proposed algorithm is a lossless and distributed pendant to CWB on the merged data.
- Instead of benchmarking the algorithm, we compared it with a GAMM fitted with `mgcv`.





Summary and outlook

Summary:

- CWB can be fit to distributed data in a lossless fashion by just relying on aggregated data that do not reveal private information about the used data set.
- Site-specific effects can be estimated by making use of the structure of tensor product base learner.
- The SSE calculation is also done easily ba sharing the SSE per site and aggregating them.

Outlook:

- Account for vertically and horizontally distributed data.
- Reduce the communication costs to speed up the fitting process.

Automation

About

- About
- Publication

Summary / Outlook

Distributed model evaluation

About

- About
- Publication

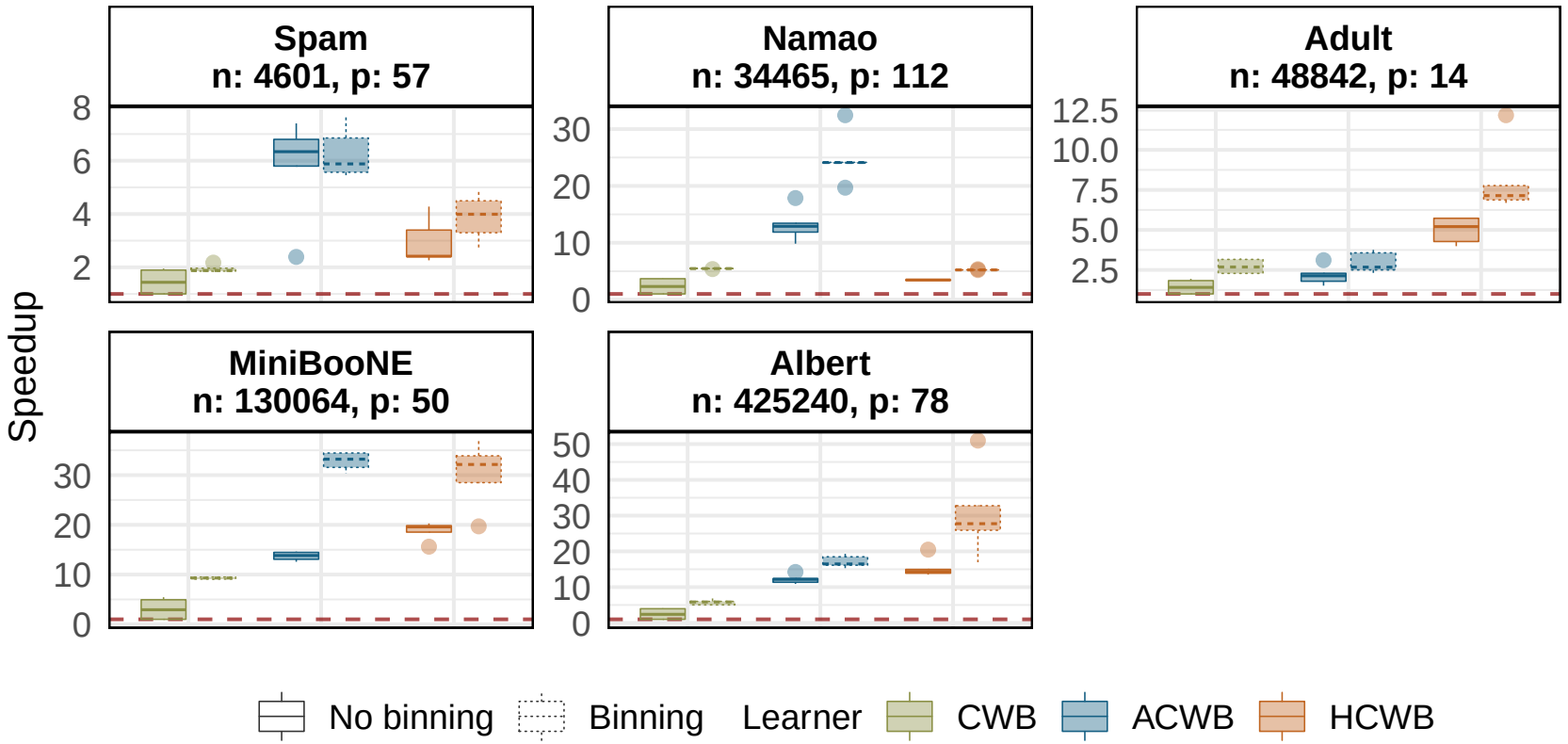
compboost

About

Was macht compboost, ganz kleines Bsp.

- About
- Publication

Speedup compared to mboost



Summary / Outlook

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115.

Audet, C. and Hare, W. (2017). *Derivative-free and blackbox optimization*, volume 2. Springer.

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Van der Vorst, H. (1994). *Templates for the solution of linear systems: building blocks for iterative methods*. SIAM.

Bates, D., Maechler, M., and Jagan, M. (2022). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.5-1.

Bekkerman, R., Bilenko, M., and Langford, J. (2011). *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press.

Biau, G., Cadre, B., and Rouvière, L. (2019). Accelerated gradient boosting. *Machine Learning*, 108(6):971–992.

References ii

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., et al. (2021). Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *arXiv preprint arXiv:2107.05847*.

Bischl, B., Mersmann, O., Trautmann, H., and Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary computation*, 20:249–75.

Bost, R., Popa, R. A., Tu, S., and Goldwasser, S. (2014). Machine learning classification over encrypted data. Cryptology ePrint Archive, Paper 2014/331. `https://eprint.iacr.org/2014/331`.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Brier, G. W. et al. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Brockhaus, S., Rügamer, D., and Greven, S. (2020). Boosting functional regression models with fdboost. *Journal of Statistical Software*, 94(10):1–50.

Browne, M. W. (2000). Cross-validation methods. *Journal of Mathematical Psychology*, 44(1):108–132.

Bühlmann, P., Hothorn, T., et al. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical science*, 22(4):477–505.

Bühlmann, P. and Yu, B. (2003). Boosting with the L2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339.

Buluc, A. and Gilbert, J. R. (2008). Challenges and advances in parallel sparse matrix-matrix multiplication. In *2008 37th International Conference on Parallel Processing*, pages 503–510.

References iii

Casalicchio, G. (2019). *On benchmark experiments and visualization methods for the evaluation and interpretation of machine learning models*. PhD dissertation, LMU Munich.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4.

Chen, Y.-R., Rezapour, A., and Tzeng, W.-G. (2018). Privacy-preserving ridge regression on distributed data. *Information Sciences*, 451:34–49.

Choi, J., Walker, D. W., and Dongarra, J. J. (1994). Pumma: Parallel universal matrix multiplication algorithms on distributed memory concurrent computers. *Concurrency: Practice and Experience*, 6(7):543–570.

Coors, S., Schalk, D., Bischl, B., and Rügamer, D. (2021). Automatic componentwise boosting: An interpretable automl system. *ECML-PKDD Workshop on Automating Data Science*.

Cunha, M., Mendes, R., and Vilela, J. P. (2021). A survey of privacy-preserving mechanisms for heterogeneous data types. *Computer Science Review*, 41:100403.

Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55.

Davis, T. A. (2006). *Direct methods for sparse linear systems*. SIAM.

DeLong, E. R., DeLong, D. M., and Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845.

References iv

Drozdal, J., Weisz, J., Wang, D., Dass, G., Yao, B., Zhao, C., Muller, M., Ju, L., and Su, H. (2020). Trust in automl: Exploring information needs for establishing trust in automated machine learning systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, page 297–307, New York, NY, USA. Association for Computing Machinery.

Duff, I. S., Grimes, R. G., and Lewis, J. G. (1989). Sparse matrix test problems. *ACM Transactions on Mathematical Software (TOMS)*, 15(1):1–14.

Dwork, C. (2006). Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407.

Eilers, P. H. and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical science*, pages 89–102.

Fang, H. and Qian, Q. (2021). Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4).

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

References v

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. In *Automated machine learning*, pages 3–33. Springer, Cham.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28.

Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge university press.

Freitas, A. A. (2019). Automated machine learning for studying the trade-off between predictive accuracy and interpretability. In Holzinger, A., Kieseberg, P., Tjoa, A. M., and Weippl, E., editors, *Machine Learning and Knowledge Extraction*, pages 48–66, Cham. Springer International Publishing.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Gambs, S., Kégl, B., and Aïmeur, E. (2007). Privacy-preserving boosting. *Data Mining and Knowledge Discovery*, 14(1):131–170.

Gaye, A., Marcon, Y., Isaeva, J., LaFlamme, P., Turner, A., Jones, E. M., Minion, J., Boyd, A. W., Newby, C. J., Nuotio, M.-L., et al. (2014). Datashield: taking the analysis to the data, not the data to the analysis. *International journal of epidemiology*, 43(6):1929–1944.

References vi

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178.

Gong, M., Xie, Y., Pan, K., Feng, K., and Qin, A. (2020). A survey on differentially private machine learning [review article]. *IEEE Computational Intelligence Magazine*, 15(2):49–64.

Gordon, D. F. and Desjardins, M. (1995). Evaluation and selection of biases in machine learning. *Machine learning*, 20(1):5–22.

Hastie, T. J. (2017). Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge.

Hofner, B., Hothorn, T., Kneib, T., and Schmid, M. (2011). A framework for unbiased model selection based on boosting. *Journal of Computational and Graphical Statistics*, 20(4):956–971.

Hofner, B., Mayr, A., and Schmid, M. (2016). gamboostLSS: An R package for model building and variable selection in the GAMLSS framework. *Journal of Statistical Software*, 74(1).

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

Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M., and Hofner, B. (2020). *mboost: Model-based boosting*. R package version 2.9-7.

Hutter, F., Kotthoff, L., and Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature.

Jayaraman, B. and Evans, D. (2019). Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912.

References vii

John, G. H. (1995). Robust decision trees: Removing outliers from databases. In *KDD*, volume 95, pages 174–179.

Karr, A. F., Lin, X., Sanil, A. P., and Reiter, J. P. (2005). Secure regression on distributed databases. *Journal of Computational and Graphical Statistics*, 14(2):263–279.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2017). Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 18(25):1–5.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2019). Auto-weka: Automatic model selection and hyperparameter optimization in weka. In *Automated machine learning*, pages 81–95. Springer, Cham.

Lang, S., Umlauf, N., Wechselberger, P., Harttgen, K., and Kneib, T. (2014). Multilevel structured additive regression. *Statistics and Computing*, 24(2):223–238.

Lazarevic, A. and Obradovic, Z. (2001). The distributed boosting algorithm. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 311–316.

Li, J., Kuang, X., Lin, S., Ma, X., and Tang, Y. (2020). Privacy preservation for machine learning training and classification based on homomorphic encryption schemes. *Information Sciences*, 526:166–179.

References viii

Li, Y., Jiang, X., Wang, S., Xiong, H., and Ohno-Machado, L. (2016). Vertical grid logistic regression (vertigo). *Journal of the American Medical Informatics Association*, 23(3):570–579.

Li, Z. and Wood, S. N. (2020). Faster model matrix crossproducts for large generalized linear models with discretized covariates. *Statistics and Computing*, 30(1):19–25.

Liew, B. X., Rügamer, D., Abichandani, D., and De Nunzio, A. M. (2020a). Classifying individuals with and without patellofemoral pain syndrome using ground force profiles – Development of a method using functional data boosting. *Gait & Posture*, 80:90–95.

Liew, B. X., Rügamer, D., Stocker, A., and De Nunzio, A. M. (2020b). Classifying neck pain status using scalar and functional biomechanical variables – Development of a method using functional data boosting. *Gait & posture*, 76:146–150.

Liu, W. and Vinter, B. (2014). An efficient gpu general sparse matrix-matrix multiplication for irregular data. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 370–381.

Lu, H., Karimireddy, S. P., Ponomareva, N., and Mirrokni, V. (2020). Accelerating gradient boosting machines. In *International Conference on Artificial Intelligence and Statistics*, pages 516–526. PMLR.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

References ix

Luo, C., Islam, M., Sheils, N. E., Buresh, J., Reps, J., Schuemie, M. J., Ryan, P. B., Edmondson, M., Duan, R., Tong, J., et al. (2022). Dlmm as a lossless one-shot algorithm for collaborative multi-site distributed linear mixed models. *Nature Communications*, 13(1):1–10.

Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. (2007). l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

Mohassel, P. and Zhang, Y. (2017). Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $O(1/k^2)$.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Pepe, M. S. (2000). An interpretation for the roc curve and inference using glm procedures. *Biometrics*, 56(2):352–359.

Pepe, M. S. (2003). The statistical evaluation of medical tests for classification and prediction. *Journal of the American Statistical Association*.

References x

Pfisterer, F. (2022). *Democratizing Machine Learning – Contributions in AutoML and Fairness*. PhD thesis, LMU Munich.

Pfisterer, F., Thomas, J., and Bischl, B. (2019). Towards human centered automl. *arXiv preprint arXiv:1911.02391*.

Prasser, F., Kohlbacher, O., Mansmann, U., Bauer, B., and Kuhn, K. A. (2018). Data integration for future medicine (difuture). *Methods Inf Med*, 57(S01):e57–e65.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Rügamer, D., Brockhaus, S., Gentsch, K., Scherer, K., and Greven, S. (2018). Boosting factor-specific functional historical models for the detection of synchronization in bioelectrical signals. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(3):621–642.

Saintigny, P., Zhang, L., Fan, Y.-H., El-Naggar, A. K., Papadimitrakopoulou, V. A., Feng, L., Lee, J. J., Kim, E. S., Hong, W. K., and Mao, L. (2011). Gene expression profiling predicts the development of oral cancer. *Cancer Prevention Research*, 4(2):218–229.

Samarati, P. and Sweeney, L. (1998). Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.

Sanderson, C. and Curtin, R. (2016). Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 1(2):26.

Sanderson, C. and Curtin, R. (2018). A user-friendly hybrid sparse matrix class in c++. In *International Congress on Mathematical Software*, pages 422–430. Springer.

Schalk, D., Bischl, B., and Rügamer, D. (2022a). Accelerated componentwise gradient boosting using efficient data representation and momentum-based optimization. *Journal of Computational and Graphical Statistics*.

Schalk, D., Bischl, B., and Rügamer, D. (2023a). Privacy-preserving and lossless distributed estimation of high-dimensional generalized additive mixed models. *arXiv preprint arXiv:2210.07723*.

Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2022b). Distributed non-disclosive validation of predictive models by a modified roc-glm. *arXiv preprint arXiv:2203.10828*.

Schalk, D., Hoffmann, V. S., Bischl, B., and Mansmann, U. (2023b). dsBinVal: Conducting distributed roc analysis using datashield. *Journal of Open Source Software*, 8(82):4545.

References xii

Schalk, D., Thomas, J., and Bischl, B. (2018). compboost: Modular framework for component-wise boosting. *Journal of Open Source Software*, 3(30):967.

Schmid, M. and Hothorn, T. (2008). Boosting additive models using component-wise p-splines. *Computational Statistics & Data Analysis*, 53(2):298–311.

Shahnaz, R., Usman, A., and Chughtai, I. R. (2005). Review of storage techniques for sparse matrices. In *2005 Pakistan Section Multitopic Conference*, pages 1–7.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89.

Sun, X., Zhang, P., Liu, J. K., Yu, J., and Xie, W. (2020). Private machine learning classification based on fully homomorphic encryption. *IEEE Transactions on Emerging Topics in Computing*, 8(2):352–364.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570.

Thomas, J., Coors, S., and Bischl, B. (2018). Automatic gradient boosting. *ICML AutoML Workshop*.

Thomas, J., Hepp, T., Mayr, A., and Bischl, B. (2017). Probing for sparse and fast variable selection with model-based boosting. *Computational and mathematical methods in medicine*, 2017.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855.

Tutz, G. and Gertheiss, J. (2016). Regularized regression for categorical data. *Statistical Modelling*, 16(3):161–200.

Van Buuren, S. (2018). *Flexible imputation of missing data*. CRC press.

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J. S. (2020). A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2):1–33.

Vuk, M. and Curk, T. (2006). Roc curve, lift chart and calibration plot. *Metodoloski zvezki*, 3(1):89.

Wang, Q. and Kurz, D. (2022). Reconstructing training data from diverse ml models by ensemble inversion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2909–2917.

Wood, S. N. (2017). *Generalized additive models: an introduction with R*. Chapman and Hall/CRC.

Wood, S. N., Li, Z., Shaddick, G., and Augustin, N. H. (2017). Generalized additive models for gigadata: Modeling the u.k. black smoke network daily data. *Journal of the American Statistical Association*, 112(519):1199–1210.

Xanthopoulos, I., Tsamardinos, I., Christophides, V., Simon, E., and Salinger, A. (2020). Putting the human back in the automl loop. In *EDBT/ICDT Workshops*.

Yan, Z., Zachrison, K. S., Schwamm, L. H., Estrada, J. J., and Duan, R. (2022). Fed-glmm: A privacy-preserving and computation-efficient federated algorithm for generalized linear mixed models to analyze correlated electronic health records data. *medRxiv*.

Zhu, R., Jiang, C., Wang, X., Wang, S., Zheng, H., and Tang, H. (2020). Privacy-preserving construction of generalized linear mixed model for biomedical computation. *Bioinformatics*, 36(Supplement_1):i128–i135.

Backup

Backup