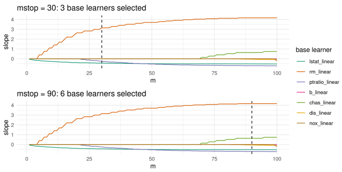
Introduction to Machine Learning

Boosting
Gradient Boosting: Advanced CWB

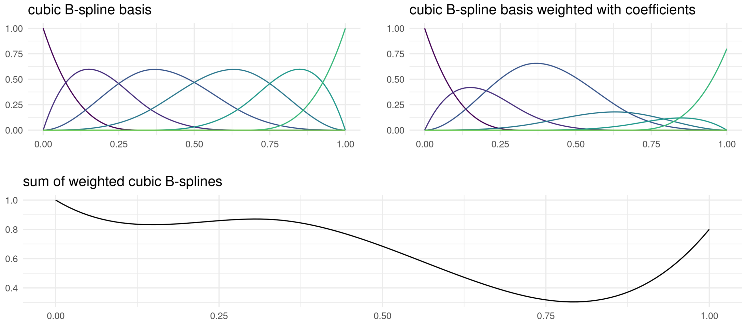


Learning goals

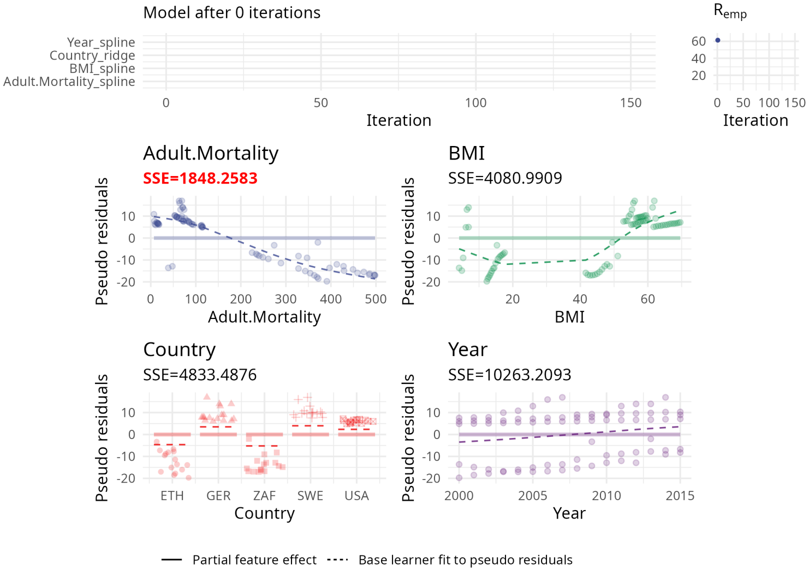
- Details of nonlinear BLs and splines
- Decomposition for splines
- Fair base learner selection
- Feature importance and PDPs

NONLINEAR BASE LEARNERS

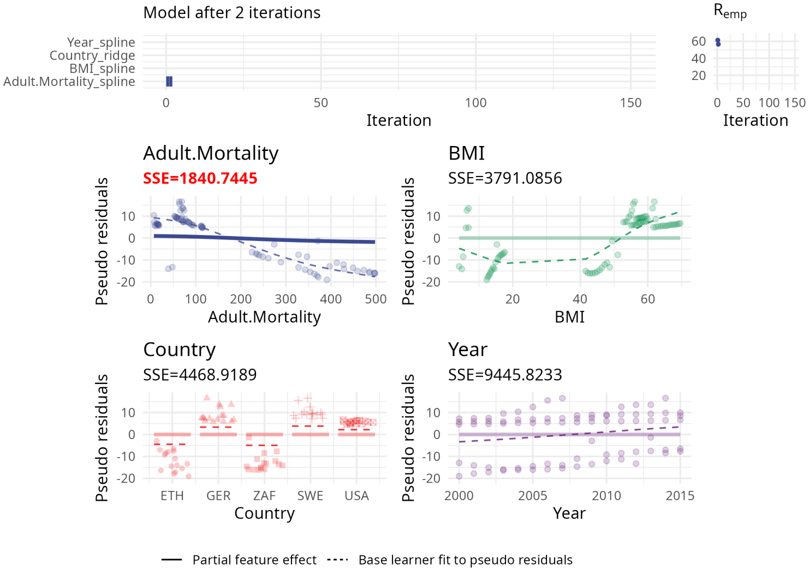
As an alternative we can use nonlinear base learners, such as *P*- or *B*-splines, which make the model equivalent to a **generalized additive model (GAM)** (as long as the base learners keep their additive structure, which is the case for splines).

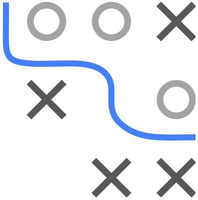


EXAMPLE: LIFE EXPECTANCY (NONLINEAR)

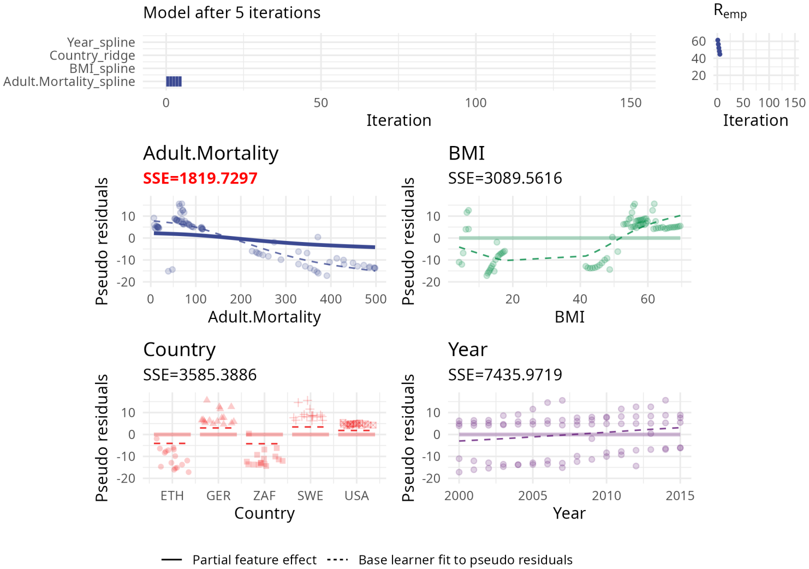


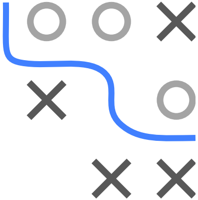
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



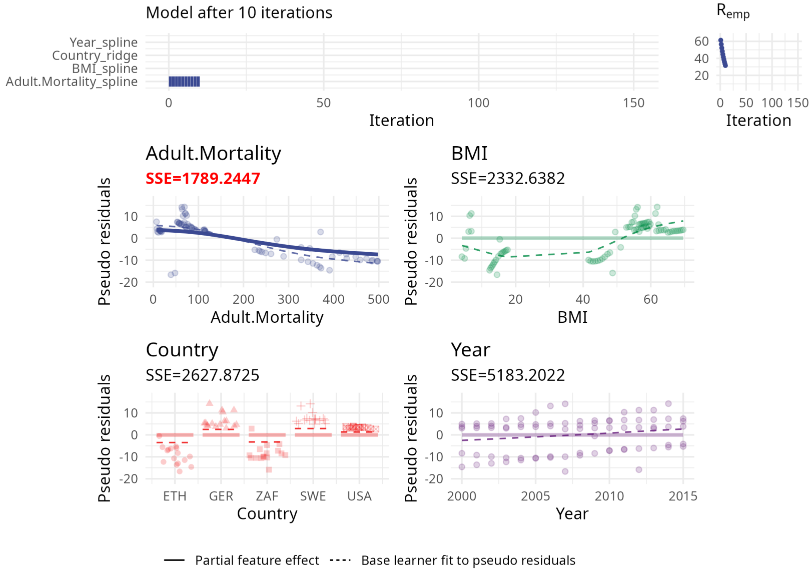


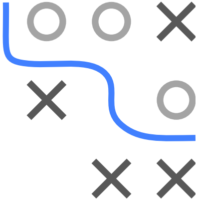
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



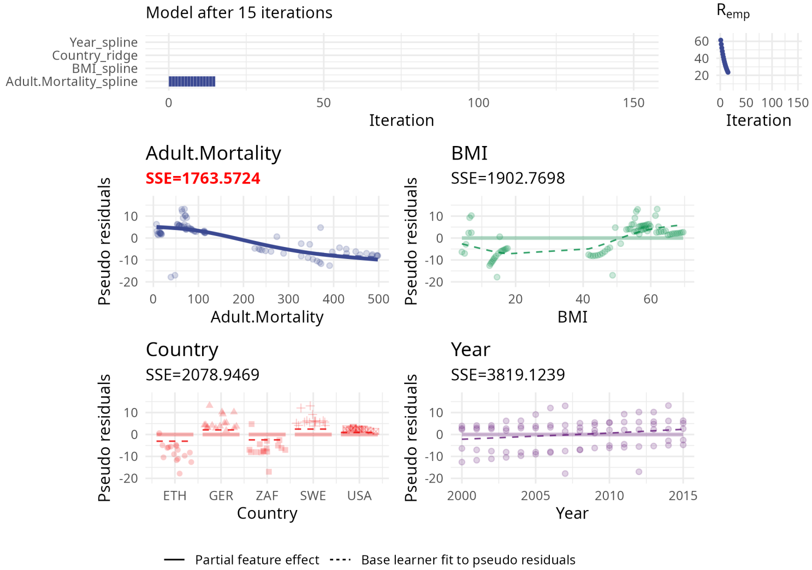


EXAMPLE: LIFE EXPECTANCY (NONLINEAR)

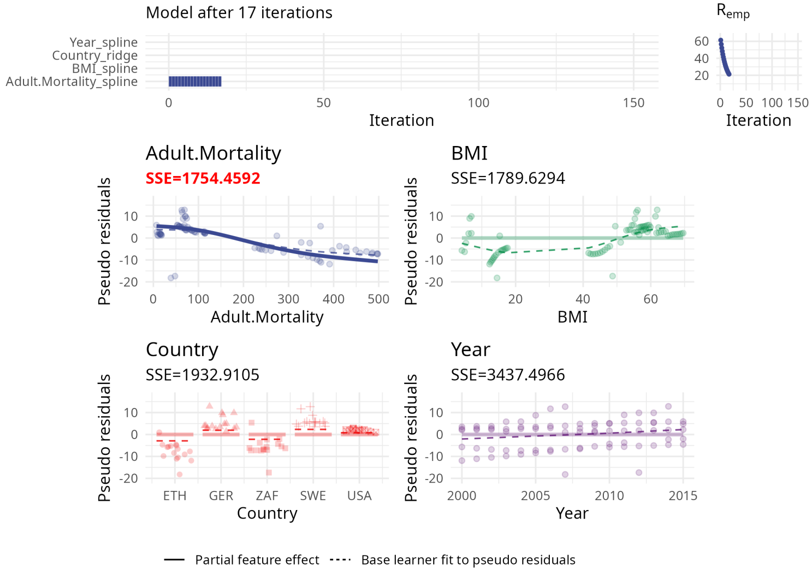


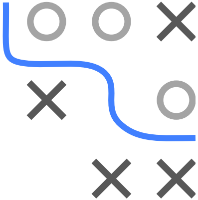


EXAMPLE: LIFE EXPECTANCY (NONLINEAR)

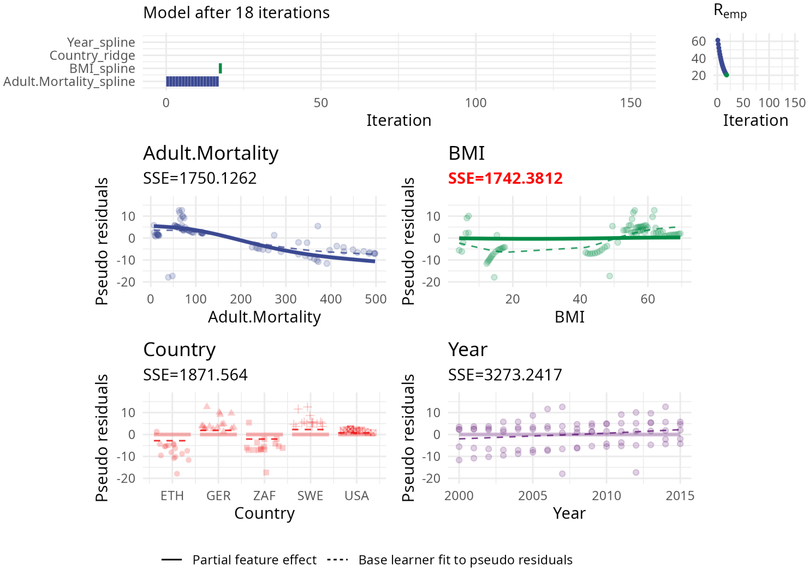


EXAMPLE: LIFE EXPECTANCY (NONLINEAR)

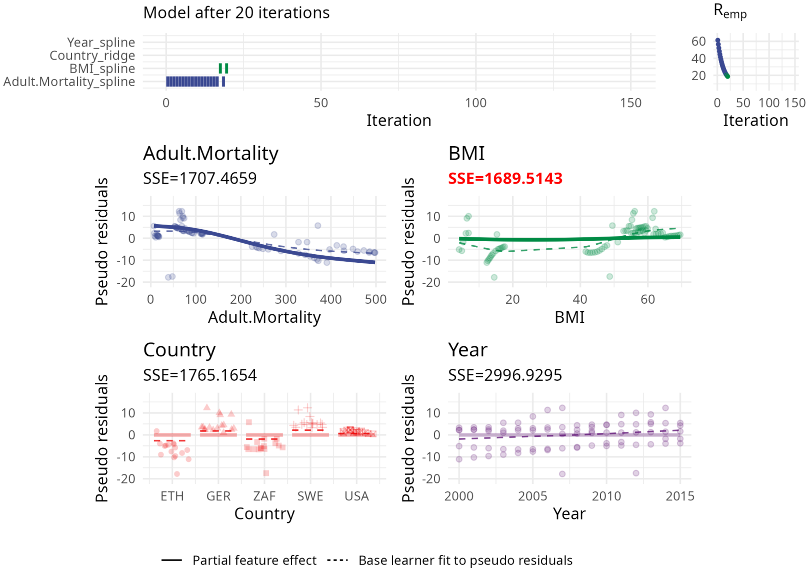




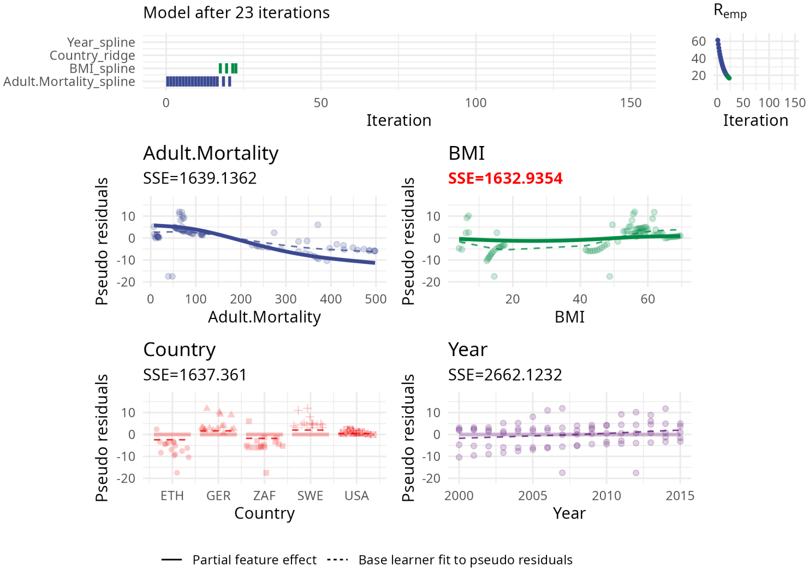
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



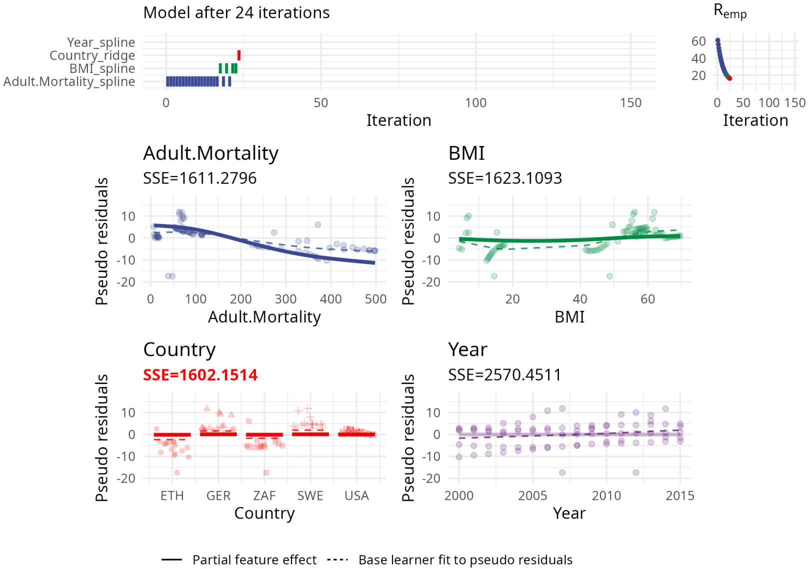
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



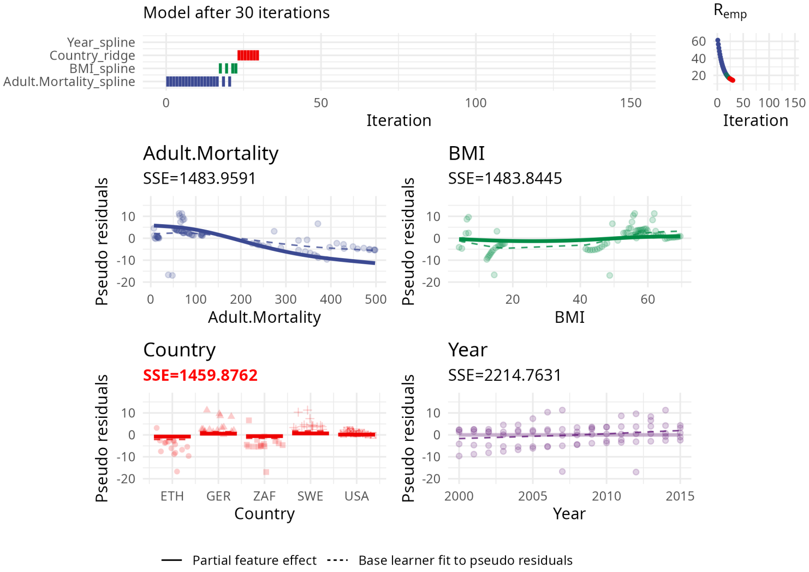
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



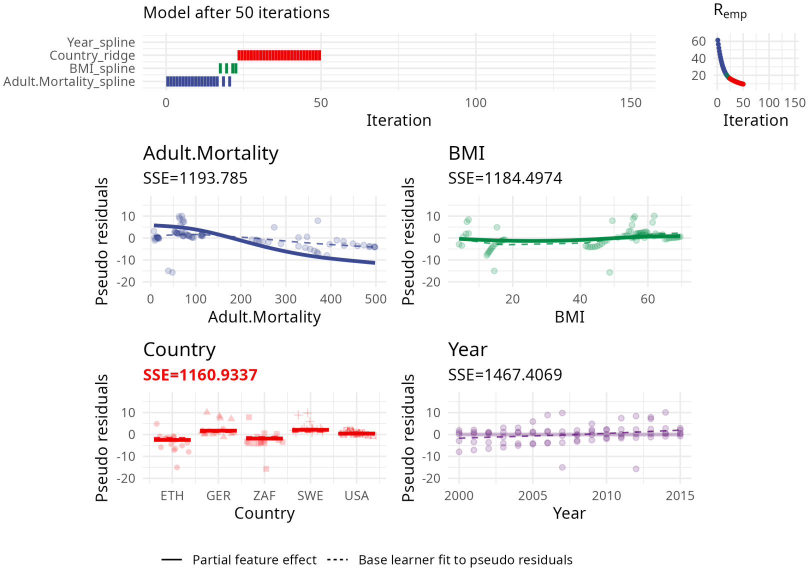
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



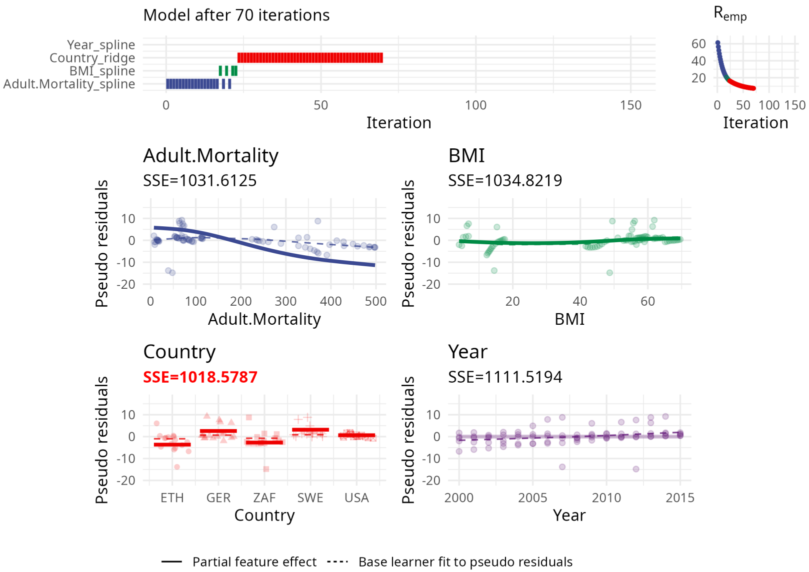
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



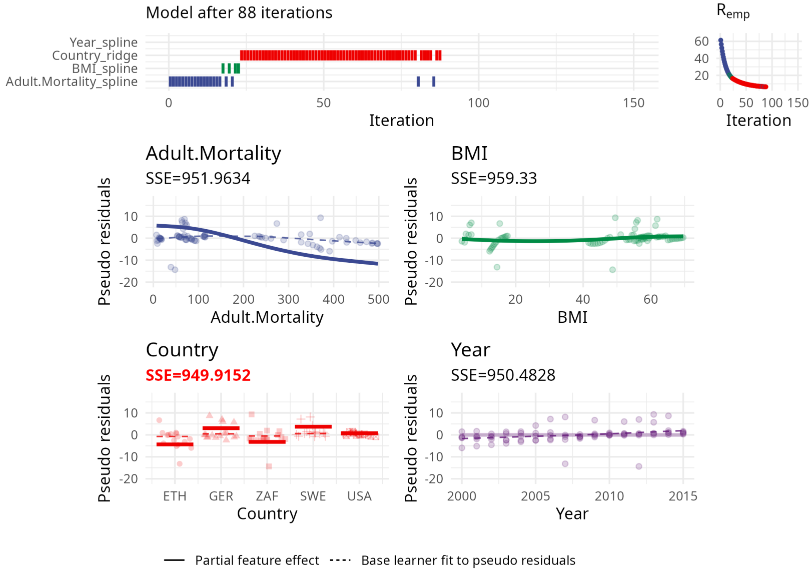
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)

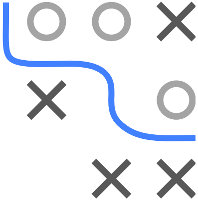


EXAMPLE: LIFE EXPECTANCY (NONLINEAR)

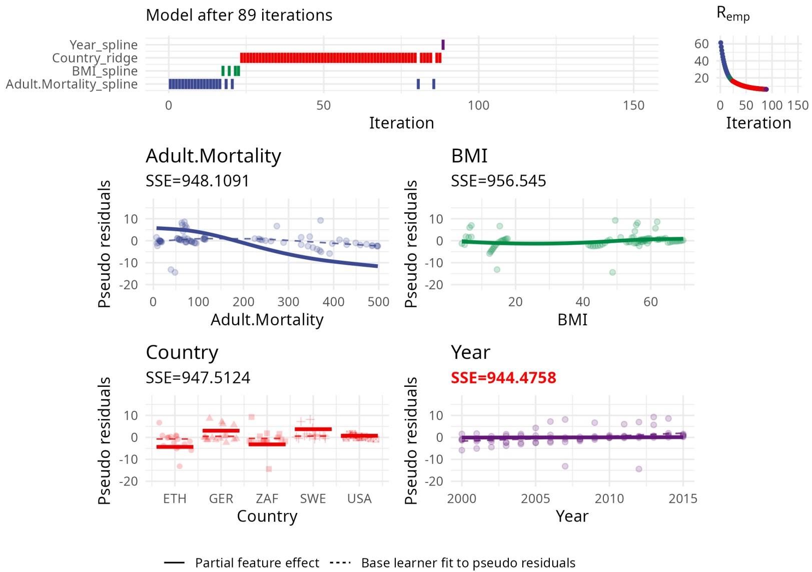


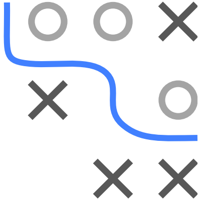
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



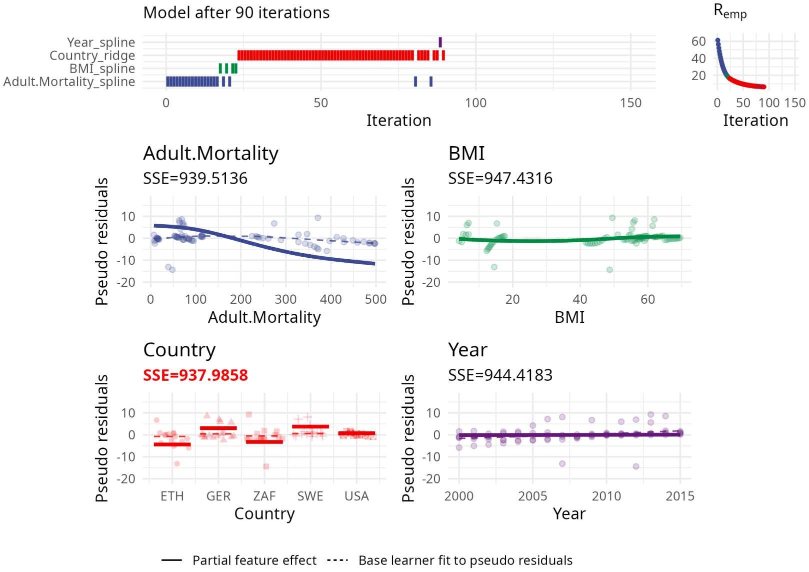


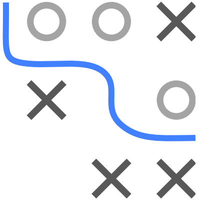
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



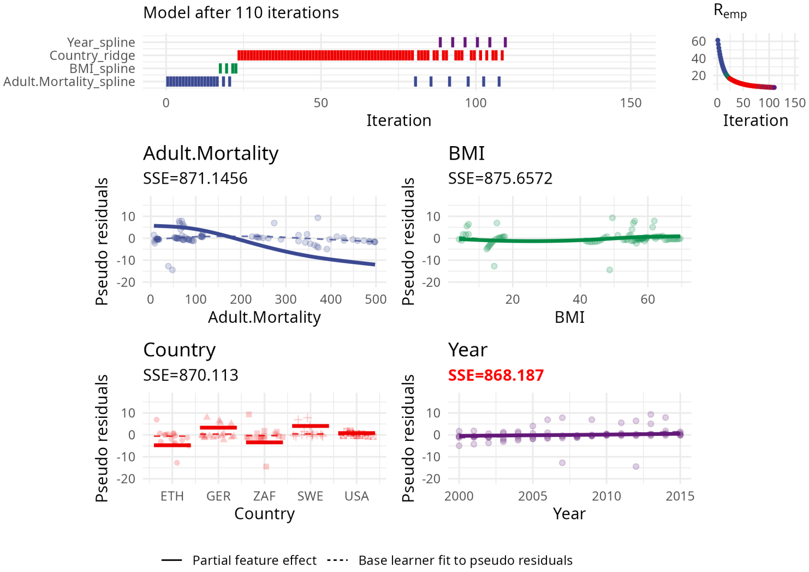


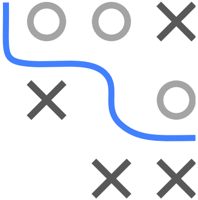
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



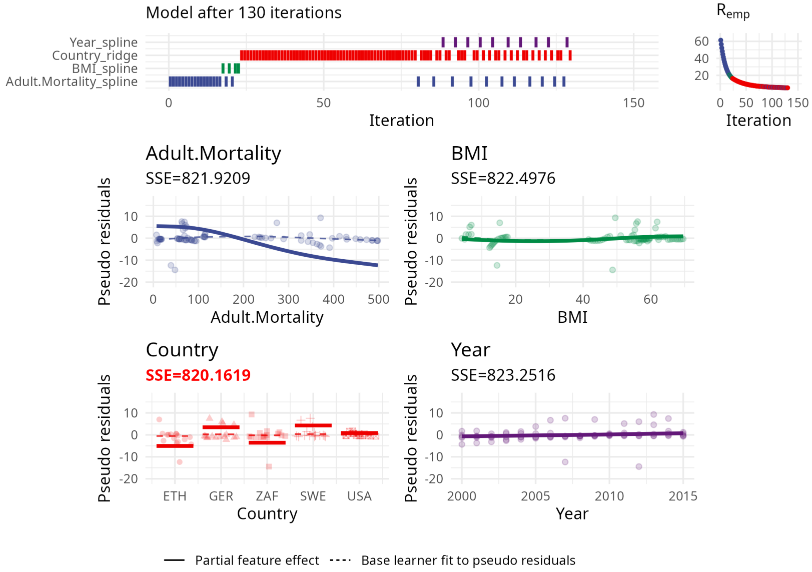


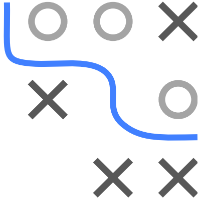
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)



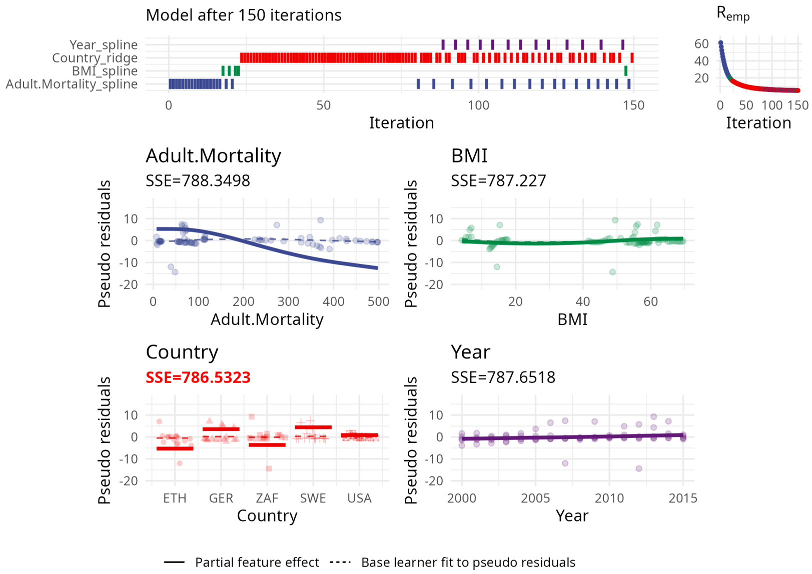


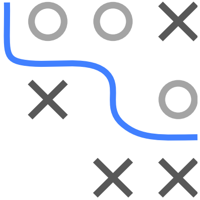
EXAMPLE: LIFE EXPECTANCY (NONLINEAR)





EXAMPLE: LIFE EXPECTANCY (NONLINEAR)





NONLINEAR EFFECT DECOMPOSITION

▸ Kneib, Hothorn, and Tutz 2009 proposed a decomposition of each base learner into a constant, a linear and a nonlinear part. The boosting algorithm will automatically decide which feature to include – linear, nonlinear, or none at all:

$$
\begin{aligned}
b_j(x_j, \boldsymbol{\theta}^{[m]}) &= b_{j,\text{const}}(x_j, \boldsymbol{\theta}^{[m]}) + b_{j,\text{lin}}(x_j, \boldsymbol{\theta}^{[m]}) + b_{j,\text{nonlin}}(x_j, \boldsymbol{\theta}^{[m]}) \\
&= \theta^{[m]}_{\text{j,const}} + x_j \cdot \theta^{[m]}_{j,\text{lin}} + s_j(x_j, \boldsymbol{\theta}^{[m]}_{j,\text{nonlin}}),
\end{aligned}
$$

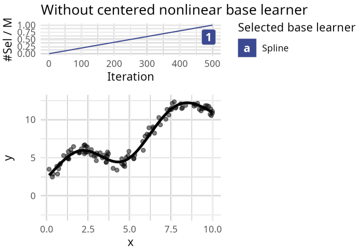
where

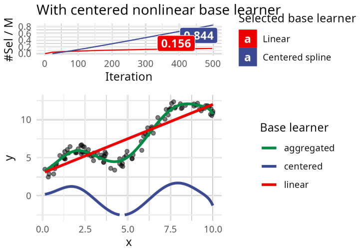
- $\theta_{\text{j,const}}$ is the intercept of feature j,
- $x_j \cdot \theta^{[m]}_{j,\text{lin}}$ is a feature-specific linear base learner, and
- $s_j(x_j, \boldsymbol{\theta}^{[m]}_{j,\text{nonlin}})$ is a (centered) nonlinear base learner capturing deviation from the linear effect

Careful: We usually also apply an orthogonalization procedure on top of this but skip technical details here.

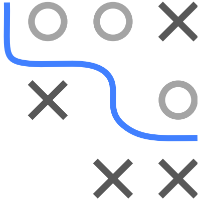
NONLINEAR EFFECT DECOMPOSITION

- Suppose $n = 100$ uniformly distributed x values between 0 and 10.
- The response $y = 2\sin(x) + x + 2 + \varepsilon$ has a nonlinear and linear component ($\varepsilon \sim \mathcal{N}(0, \frac{1}{2})$).
- We apply CWB with $M = 500$ to $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ with:
  - One model with $\mathcal{B} = \{b_{j,\text{lin}}, b_{j,\text{nonlin}}\}$
  - One model with $\mathcal{B} = \{b_{j,\text{lin}}, b_{j,\text{nonlin}^c}\}$



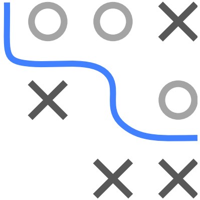


FAIR BASE LEARNER SELECTION



- Using splines and linear base learners in CWB will favor the more complex spline BLs over the linear BLs
- This makes it harder to achieve the desired behavior of the base learner decomposition as explained previously
- To conduct a fair base learner selection, we set the degrees of freedom of all base learners equal
- The idea is to set a single learner's regularization/penalty term so that their complexity is treated equally
- We also skip some technical details here

AVAILABLE BASE LEARNERS



There is a large number of possible base learners, e.g.:

- Linear effects and interactions (with or without intercept)
- Uni- or multivariate splines and tensor product splines
- Trees
- Random effects and Markov random fields
- Effects of functional covariates
- ...

In combination with the flexible choice of loss functions, boosting can be applied to fit a huge class of models.

Recent extensions include distributional regression (GAMLSS), where multiple additive predictors are boosted to model all distributional parameters (e.g., cond. mean and variance for a Gaussian model).

PARTIAL DEPENDENCE PLOTS

If we use single features in base learners, we consider each BL as a wrapper around a feature representing the feature's effect on the target. BLs can be selected more than once (with varying parameter estimates), signaling that this feature is more important.
E.g. let $j \in \{1, 2, 3\}$, the first three iterations might look as follows

$$
\begin{aligned}
m = 1: &\quad \hat{f}^{[1]}(\mathbf{x}) = \hat{f}^{[0]} + \alpha \hat{b}_2(x_2, \hat{\theta}^{[1]}) \\
m = 2: &\quad \hat{f}^{[2]}(\mathbf{x}) = \hat{f}^{[1]} + \alpha \hat{b}_3(x_3, \hat{\theta}^{[2]}) \\
m = 3: &\quad \hat{f}^{[3]}(\mathbf{x}) = \hat{f}^{[2]} + \alpha \hat{b}_2(x_2, \hat{\theta}^{[3]})
\end{aligned}
$$

Due to linearity, $\hat{b}_2$ base learners can be aggregated:

$$\hat{f}^{[3]}(\mathbf{x}) = \hat{f}^{[0]} + \alpha\left(\hat{b}_2(x_2, \hat{\theta}^{[1]} + \hat{\theta}^{[3]}) + \hat{b}_3(x_3, \hat{\theta}^{[2]})\right)$$

Which is equivalent to: $\hat{f}^{[3]}(\mathbf{x}) = \hat{f}_0 + \hat{f}_2(x_2) + \hat{f}_3(x_3)$.
Hence, $\hat{f}$ can be decomposed into the marginal feature effects (PDPs).

FEATURE IMPORTANCE

- We can further exploit the additive structure of the boosted ensemble to compute measures of **variable importance**.
- To this end, we simply sum for each feature x_j the improvements in empirical risk achieved over all iterations until $1 < m_{\text{stop}} \leq M$:

$$VI_j = \sum_{m=1}^{m_{\text{stop}}} \left(\mathcal{R}_{\text{emp}} \left(f^{[m-1]}(\mathbf{x}) \right) - \mathcal{R}_{\text{emp}} \left(f^{[m]}(\mathbf{x}) \right) \right) \cdot \mathbb{I}_{[j \in j^{[m]})]},$$

TAKE-HOME MESSAGE

- Componentwise gradient boosting is the statistical re-interpretation of gradient boosting
- We can fit a large number of statistical models, even in high dimensions ($p \gg n$)
- A drawback compared to statistical models is that we do not get valid inference for coefficients $\rightarrow$ post-selection inference
- In most cases, gradient boosting with trees will dominate componentwise boosting in terms of performance due to its inherent ability to include higher-order interaction terms