An introduction to boosting GAMLSS with applications in **R**

Andreas Mayr

Department of Medical Informatics, Biometry and Epidemiology
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

8 May 2013
University of Essex

The GAMLSS model class

Generalized Additive Models for Location, Scale and Shape

$$Y \sim D(\mu, \sigma, \nu, \tau)$$

$$g_\mu(\mu) = \eta_\mu = \beta_{0\mu} + \sum_{j=1}^{p_\mu} f_{j\mu}(x_j) \quad \text{“location”}$$

$$g_\sigma(\sigma) = \eta_\sigma = \beta_{0\sigma} + \sum_{j=1}^{p_\sigma} f_{j\sigma}(x_j) \quad \text{“scale”}$$

$$g_\nu(\nu) = \eta_\nu = \beta_{0\nu} + \sum_{j=1}^{p_\nu} f_{j\nu}(x_j) \quad \text{“skewness”}$$

$$g_\tau(\tau) = \eta_\tau = \beta_{0\tau} + \sum_{j=1}^{p_\tau} f_{j\tau}(x_j) \quad \text{“kurtosis”}$$

Fitting algorithm: the gold standard

The gamlss package

```
gamlss(formula = y ~ pb(x), sigma.formula = NULL, nu.formula = NULL,
       tau.formula = NULL, family = NO(), data = NULL, ...)
```

- More than 70 continuous, discrete and mixed distributions available.
- Maximum (penalized) likelihood estimation via Newton-Raphson / Fisher scoring.
- Modified versions of back-fitting (as for conventional GAMs) are used for the additive terms.
- Step-wise variable selection based on information criteria available.
- Works best in low-dimensional settings.

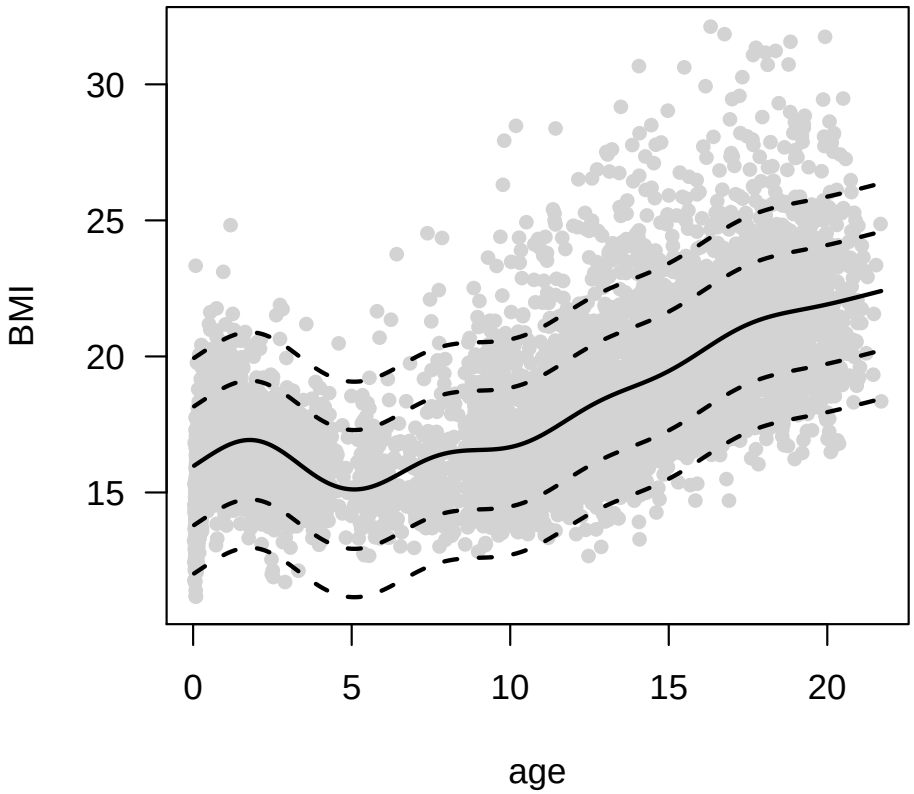
Example: Reference growth charts

- Children growth charts are used to compare the measurements of a single child to the distribution in a reference population (depending on age)
- Typically, regions are provided to judge if the development of a child is below or beyond average $\Rightarrow$ (0.03, 0.15, 0.5, 0.85, 0.97)-quantiles
- In our example we have BMI measurements of 7294 Dutch boys from birth up to the age of 20.

WHO child growth standards

The WHO recommends the usage of GAMLSS in combination with the box-cox power exponential distribution to estimate centile curves.

Example: Reference growth charts: GAM




```
> library(gamlss) ## for the data
> library(mgcv)
> data(dbbmi)  ## dutch boys BMI
> g1 <- gam(bmi ~ s(age),
+           data = dbbmi)
> ## Plot mean and quantile curves
> plot(dbbmi$age, dbbmi$bmi)
> lines(dbbmi$age, fitted(g1))
> ## Quantiles:
> q3 <- qnorm(fitted(g1),
+             sd(g1$residuals),
+             p = 0.03)
> lines(dbbmi$age, q3, lty = 2)
> ## The same for other quantiles
```

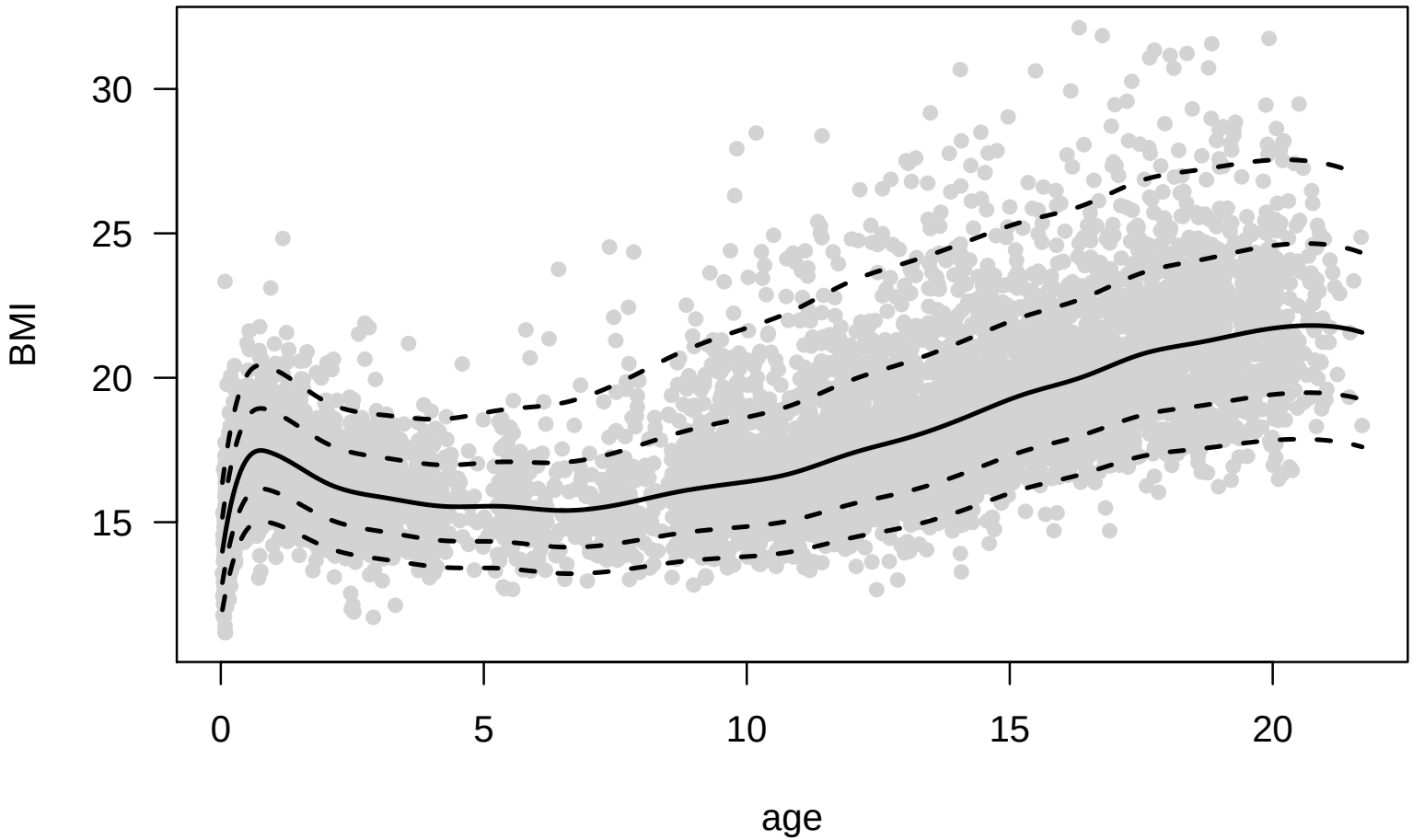
GAM

The classical mean regression model results in quantiles that are parallel to the mean.

Example: Reference growth charts – GAMLSS

```
> g2 <- gamlss(bmi ~ pb(age), sigma.formula = ~ pb(age),
+              nu.formula = ~ pb(age), tau.formula = ~ pb(age),
+              family = BCPE(), data = dbbmi)

GAMLSS-RS iteration 1: Global Deviance = 29478.27
GAMLSS-RS iteration 2: Global Deviance = 29179.41
....
GAMLSS-RS iteration 8: Global Deviance = 29166.9

> ## Plot the median
> plot(dbbmi$age, dbbmi$bmi, col = "lightgrey", pch = 19,
+      ylab="BMI", xlab="age", las = 1)
> lines(dbbmi$age, fitted(g2), lwd=2)
> ## Get the other quantiles:
> q3 <- qBCPE(p = 0.03, mu = fitted(g2), sigma = fitted(g2, "sigma"),
+             nu = fitted(g2, "nu"), tau = fitted(g2, "tau"))
> lines(dbbmi$age, q3, lwd = 2, lty = 2)
> ## The same p = c(0.15, 0.85, 0.97)
```

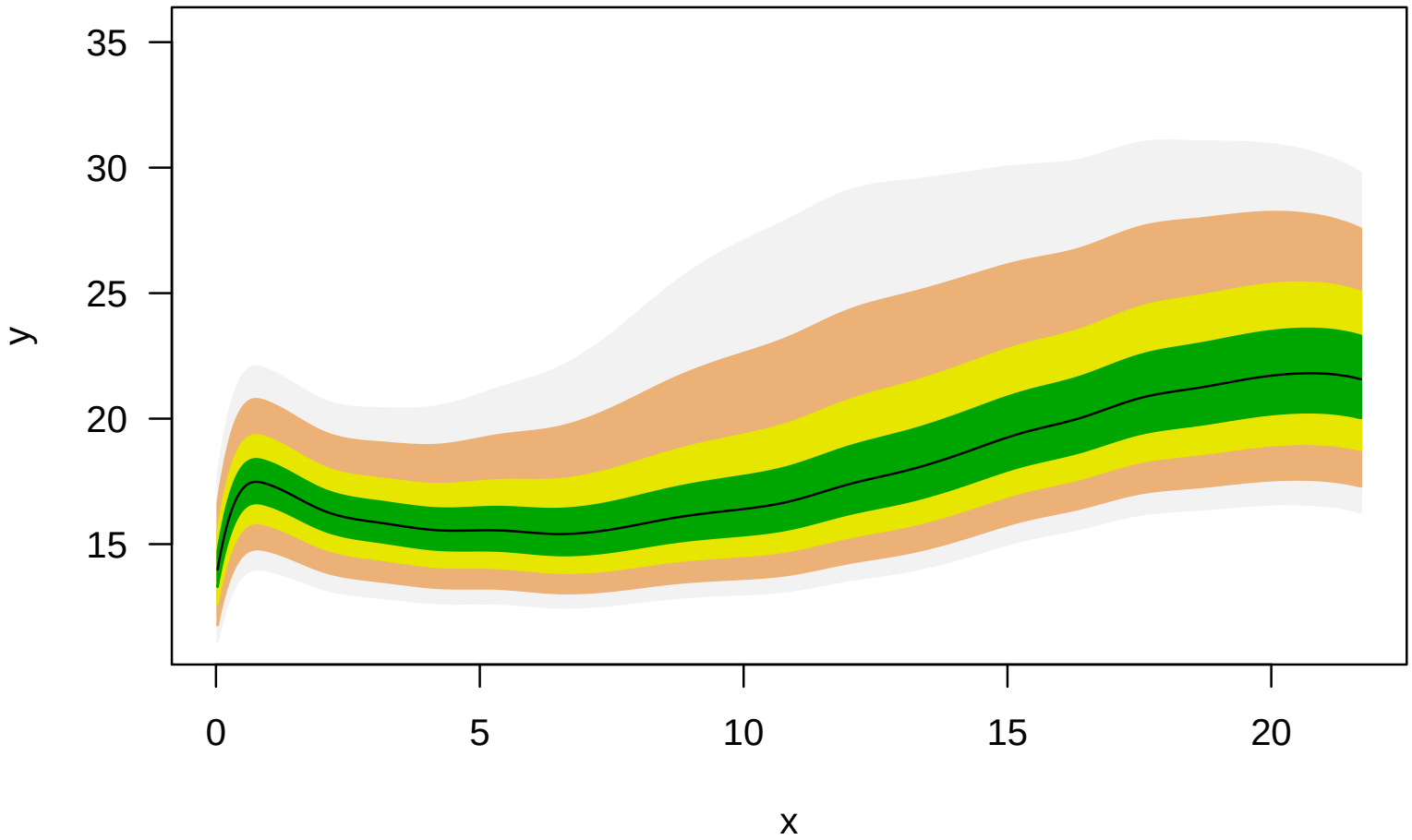
Example: Reference growth charts: GAMLSS



GAMLSS

Via GAMLSS the quantiles can report the age-specific skewness.

Example: Reference growth charts: GAMLSS




```
> ## automated centile plotting
> centiles.fan(g2, xvar = dbbmi$age, color = "terrain")
```

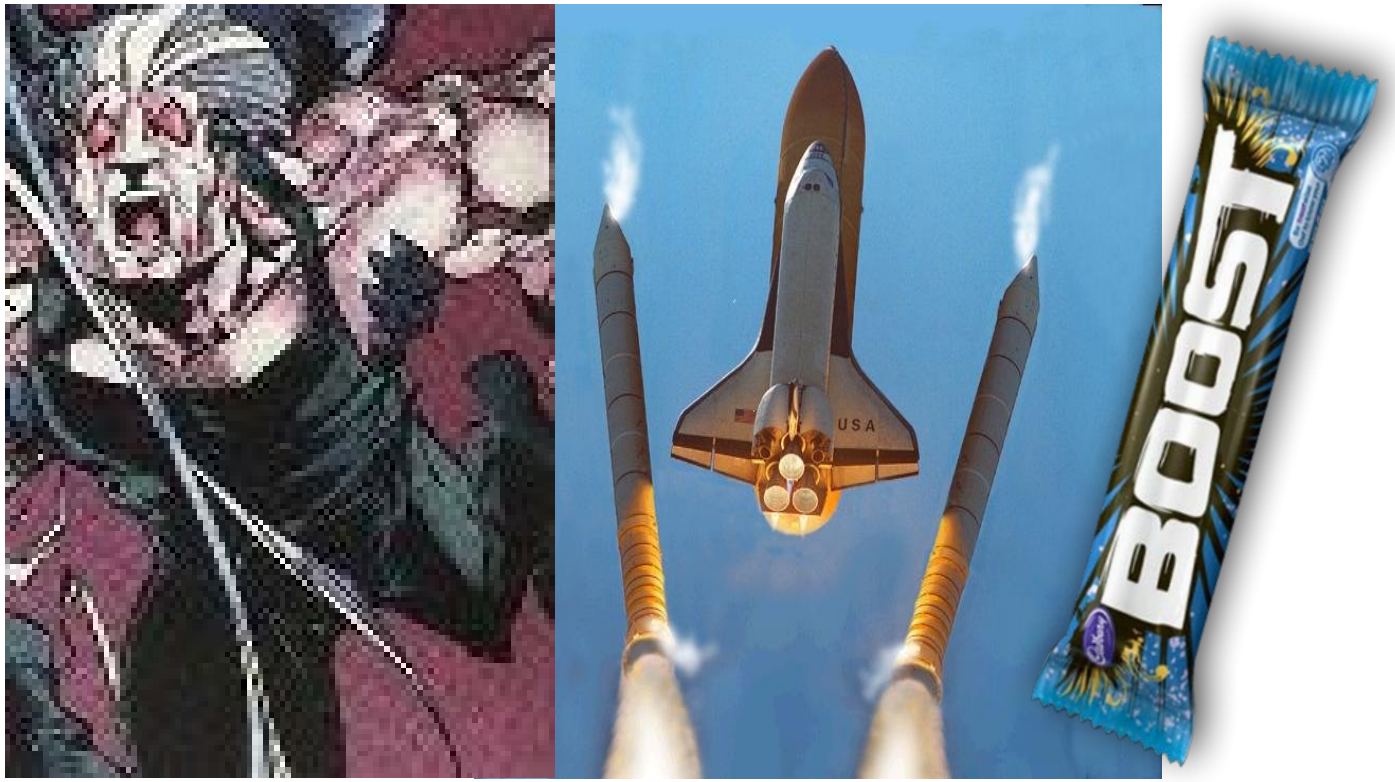
Alternative fitting scheme: boosting

The gamboostLSS package

```
gamboostLSS(formula, families = GaussianLSS(mu = NULL, sigma = NULL),
            control = boost_control(mstop = 100), ...)
```

- The fitting methods provided with with the **gamlss** package are the well tested and user-friendly gold-standard for fitting GAMLSS.
- However, in some situations it might make sense to consider boosting as an alternative:
  - Boosting works for high-dimensional settings with $p > n$.
  - Boosting is very flexible when it comes to the type of covariate effects (e.g., spatial effects)
  - Boosting includes variable selection and shrinkage of effect estimates (similar to the lasso)

Boosting ? – searching the Web



The Boost character of Marvel Comics, the solid rocket boosters separating from NASA's Space Shuttle Endeavour and a chocolate bar named Boost.

Boosting ? – searching a dictionary

boost

verb /bu:st/[T]

---

to improve or increase something

The theatre managed to boost its audiences by cutting ticket prices.

Share prices were boosted by reports of the President's recovery.

I tried to boost his ego (= make him feel more confident) *by praising his cooking.*

(Definition of the verb boost from the Cambridge Advanced Learner's Dictionary & Thesaurus, Cambridge University Press, 2012)

Machine learning

The idea of boosting emerged from the field of machine learning:

- ▷ Automated learning of an algorithm based on data observed in the past in order to make valid predictions for the future.

Typically, the machine should yield a generalization function (called hypothesis $\hat{h}(\cdot)$) for a classification problem.

- In the easiest case, outcome Y has two classes, coded as $\{-1, 1\}$.
- The machine should learn from a training sample $(y_1, \boldsymbol{x}_1), ..., (y_n, \boldsymbol{x}_n)$, where $\boldsymbol{x}_i$ are possible predictors.

$$(y_1, \boldsymbol{x}_1), ..., (y_n, \boldsymbol{x}_n) \overset{\text{machine-learning}}{\longrightarrow} \hat{h}(\boldsymbol{x}_{\text{new}}) = \hat{y}_{\text{new}}$$

The boosting question

The first boosting algorithm ever, was in fact no algorithm for practical purposes but the proof in a theoretical discussion:

Kearns, 1988

Does the existence of a *weak* learner for a certain problem imply the existence of a *strong* learner?

- **Weak** learners are defined as a prediction rule with a correct classification rate at least slightly better than random guessing.
- **Strong** learners, on the other hand, should be very close to a perfect classification.

In practice, it is often very easy to get weak learners (spam filters: search for word *Viagra*) but very hard to get strong learners.

The boosting answer

Schapire (1989), Freund (1990)

Any weak (base) learner can be iteratively boosted to become a strong learner.

- The proof to this ground-breaking idea generated the first boosting algorithm.
- To *improve* the performance of the weak base-learner, this learner is applied various times on a re-wighted data set and afterwards combined to a more powerful solution.
- **Idea:**
  - *Increase* the weights of observations hard to classify.
  - ▷ Force the algorithm to find solutions for observations that were mis-classified in previous iterations.

The boosting idea: classification

- The weak base-learner is iteratively applied to a re-wighted data set, iterations $m = 1, ..., m_{\text{stop}}$.
- Observations that were misclassified in the previous round get higher weights $w^{[m]}$.

$$
\begin{aligned}
\text{re-weighted data } \boldsymbol{y}^\top \boldsymbol{w}^{[m-1]} &\xrightarrow{\text{base-learner}} \hat{h}^{[m]}(\boldsymbol{x}) \\
\text{re-weighted data } \boldsymbol{y}^\top \boldsymbol{w}^{[m]} &\xrightarrow{\text{base-learner}} \hat{h}^{[m+1]}(\boldsymbol{x}) \\
\ldots &\quad \ldots \\
\ldots &\quad \ldots \\
\text{re-weighted data } \boldsymbol{y}^\top \boldsymbol{w}^{[m_{\text{stop}}-1]} &\xrightarrow{\text{base-learner}} \hat{h}^{[m_{\text{stop}}]}(\boldsymbol{x})
\end{aligned}
$$

The boosting idea: classification

- In the end, all solutions of the base-learner are aggregated to a final prediction.
- Solutions of the base-learner that showed a higher accuracy (in iteration m) get a higher weight α_m :

Weighted majority voting:

$$\hat{f}_{\text{boost}}(\boldsymbol{x}) = \text{sign}\left(\sum_{m=1}^{m_{\text{stop}}} \alpha_m \hat{h}^{[m]}(\boldsymbol{x})\right)$$

The authors later compared this general concept of boosting to *garnering wisdom from a council of fools* (Schapire and Freund, 2012).

The famous AdaBoost

- The general concept of boosting led to the development of AdaBoost (Freund and Schapire 1996)
- AdaBoost is typically used with classification trees or stumps as base-learner
- AdaBoost is the most popular boosting algorithm and is still today center of many methodological developments in machine learning

Breiman (the inventor of random forest):

Boosting is the best off-the-shelf classifier in the world (as cited in Hastie et al., 2009)

The statistical view of boosting

- There exist many theoretical interpretations for the success of AdaBoost (game theory, margins theory)
- For the statistical community, the most important one is the statistical view of boosting (Friedman et al., 2000)
- It laid the groundwork to apply boosting for statistical modeling.
- The authors showed that AdaBoost with regression type base-learners can be used to fit a logistic regression model.

Friedman, 2001

Boosting can be used to estimate the unknown quantities in almost every statistical regression setting. ▷ Boosting as functional gradient descent.

Statistical modelling

- In contrast to most machine learning tools, statistical models are no black-box prediction schemes but by definition interpretable

$$\text{response} = f(\text{predictors}) + \varepsilon$$

- More formally, the optimization problem for statistical modeling can be formulated as

$$\hat{f}(\cdot) = \underset{f(\cdot)}{\operatorname{argmin}} \Big\{ \mathsf{E}_{Y,X} \Big[\rho(Y, f(X)) \Big] \Big\}$$

  where $\rho(\cdot)$ is the loss function.
- The most common loss function is the L_2 loss

$$\rho(y, f(\cdot)) = (y - f(\cdot))^2$$

  leading to classical least squares regression of the mean:

$$f(\boldsymbol{x}) = \mathsf{E}(Y | X = \boldsymbol{x})$$

Boosting as functional gradient descent

- The idea behind statistical boosting is to descent the empirical risk via fitting the negative gradient of the loss function.
- For each potential predictor typically one regression-type base-learner is specified.
- The base-learner reflects the type of effect the predictor will have on the response (linear model results in linear effect; splines can result in a non-linear effect)
- The base-learners are fitted one-by-one to the negative gradient(component-wise fitting).
- In every boosting iteration, only the fit of the best performing base-learner is included in the model ▷ variable selection.

Model-based boosting

Gradient descent boosting is provided in **R** with the powerful `mboost` package (Hothorn et al., 2013)

Model-based boosting

The mboost package

```
gamboost(formula, family = Gaussian(),
         control = boost_control(mstop = 100), ...)
```

- The package provides a unified boosting framework for various regression settings.
- There exist pre-defined base-learners for linear, non-linear, spatial, random and monotonic effects.
- Various loss functions can be combined with any of those base-learners.

mboost families

	continuous response	binary response	count data	ordered response	censored data
Standard regression	`Gaussian`				
Median regression	`Laplace`				
Quantile regression	`QuantReg`				
Expectile regression	`ExprectReg`				
Robust regression	`Huber`				
Gamma regression	`GammaReg`				
Logistic regression		`Binomial`			
AdaBoost classification		`AdaExp`			
AUC regression		`AUC`			
Poisson regression			`Poisson`		
Negative binomial model			`NBinomial`		
Proportional odds model				`ProppOdds`	
Proportional hazards model					`CoxPH`
Weibull AFT model					`Weibull`
Log-logistic AFT model					`Loglog`
Log-normal AFT model					`Lognormal`

An overview on the currently implemented distributions or loss-functions which can be specified via `family` in the fitting functions of **mboost**.

mboost base-learners

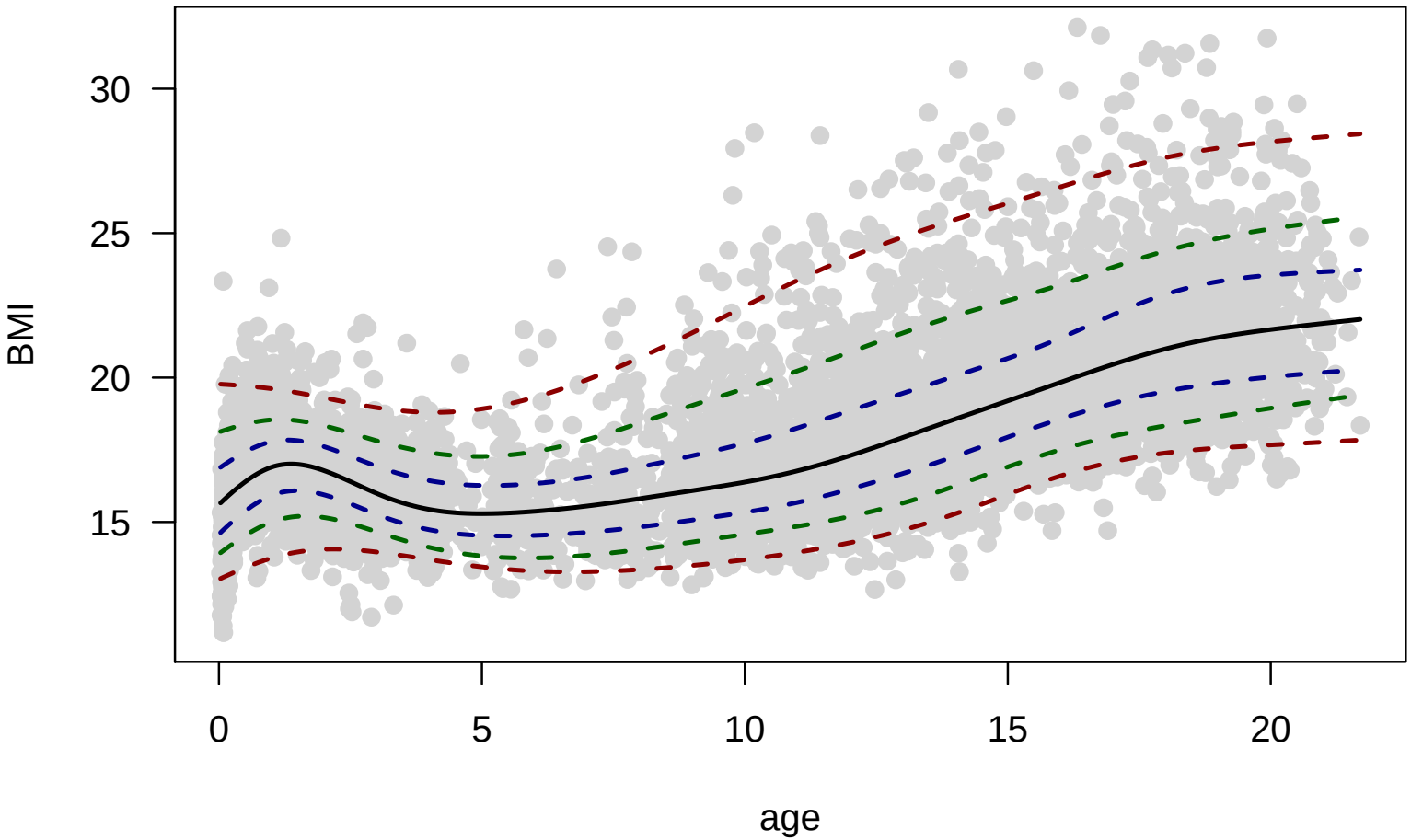
call	effect
`bols(x)`	linear effect: $\boldsymbol{x}^\top \beta$ with $\boldsymbol{x}^\top = (1, x)$
`bols(z)`	categorical effect for factor z
`bols(x, by = z)`	interaction: $\boldsymbol{x}^\top \beta \cdot z$.
`bbs(x)`	smooth effect: cubic P-splines with second order differences, 20 inner knots and 4 degrees of freedom.
`bbs(x, by = z)`	varying coefficients
`bspatial(s1, s2)`	spatial effect: tensor product P-splines
`bmrf(s)`	discrete spatial effect: Markov random field for regions s
`brandom(z)`	random intercept
`brandom(z, by = x)`	random slope
`bmono(x)`	monotonic effect: smooth effects with constraints.

An overview on the currently implemented base-learners of **mboost**.

Example: dutch boys – Quantile boosting

```
> ## Same dutch boys example as before
> ## We use QuantReg() as distribution-free loss for quantile regression
> qr05 <- gamboost(bmi ~ bbs(age), data = dbbmi, family = QuantReg(0.5),
+                  control = boost_control(mstop = 5000, trace = TRUE))

[   1] ..................................... -- risk: 5755.284
[  41] ..................................... -- risk: 5707.954

...

[4'961]  .....................................
Final risk: 5594.701

> ## The same for the other quantiles
```

Example: dutch boys – Quantile boosting



▷ Fitting the quantiles separately via boosting results in a similar plot than first fitting a GAMLSS distribution and computing the quantiles afterward.

Boosting GAMLSS

Aim: We want to adapt boosting to fit GAMLSS distributions.

- Gradient boosting can be used to optimize any loss that is at least convex and differentiable.
- However, it is only optimizing in one dimension

$$\hat{f}(\cdot) = \underset{\eta(\cdot)}{\operatorname{argmin}} \Big\{ \mathsf{E}_{Y,X} \Big[\rho(Y, \eta(X)) \Big] \Big\}$$

- Typically, $f(\cdot)$ refers to the additive predictor η for modelling the expected mean of a distribution.

▷ For GAMLSS, we have to extend the gradient boosting approach to multiple dimensions:

$$\boldsymbol{\eta} = (\eta_\mu, \eta_\sigma, \eta_\nu, \eta_\tau)$$

Optimization problem for GAMLSS

- The task is to model the distribution parameters of the conditional density $f_{\text{dens}}(y|\mu, \sigma, \nu, \tau)$
  - ▷ The optimization problem can be formulated as

$$(\hat{\mu}, \hat{\sigma}, \hat{\nu}, \hat{\tau}) = \underset{\eta_\mu, \eta_\sigma, \eta_\nu, \eta_\tau}{\operatorname{argmin}} \mathbb{E}_{Y,X}\left[\rho(Y, \eta_\mu(X), \eta_\sigma(X), \eta_\nu(X), \eta_\tau(X))\right]$$

with $\rho = -l$, the negative log-likelihood of the response distribution:

$$l = \sum_{i=1}^{n} \log\left[f_{\text{dens}}(y_i|\boldsymbol{\theta}_i)\right] = \sum_{i=1}^{n} \log\left[f_{\text{dens}}(y_i|\mu_i, \sigma_i, \nu_i, \tau_i)\right]$$

- **Idea:** We have to circle through the different dimensions of the loss.

Boosting GAMLSS: gamboostLSS

1 Initialize the additive predictors $\hat{\eta}_{\mu_i}^{[0]}, \hat{\eta}_{\sigma_i}^{[0]}, \hat{\eta}_{\nu_i}^{[0]}, \hat{\eta}_{\tau_i}^{[0]}$ with offset values, e.g. $\hat{\eta}_{\theta_{ki}}^{[0]} = 0$ for $k = 1, ..., 4$ and $i = 1, ..., n$.

2 For each distribution parameter θ_k, $k = 1, \ldots, 4$, specify a set of base-learners $h_{k1}(\cdot), \ldots, h_{kp_k}(\cdot)$ – one for each covariate. Set $m = 0$.

3 Increase m by 1.

4 (a) Set $k = 0$.

(b) Increase k by 1. If $m > m_{\text{stop,k}}$ proceed to step 4(f). Else compute the negative gradient $-\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta}_i)$ and plug in the current estimates $\boldsymbol{\eta}_i$:

$$\boldsymbol{u}_k^{[m-1]} = \left(-\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta}_i) \right)_{i=1,\ldots,n} .$$

(c) Fit the negative gradient vector $\boldsymbol{u}_k^{[m-1]}$ separately to each of the base-learners specified for the predictor η_{θ_k} in step 2.

$\vdots$

$\vdots$

(d) Select the component j^* that best fits the negative gradient vector:

$$j^* = \underset{1 \leq j \leq p_k}{\operatorname{argmin}} \sum_{i=1}^{n} (u_{ik}^{[m-1]} - h_{kj}(\cdot))^2 \ .$$

(e) Update the additive predictor η_{θ_k} as follows:

$$\hat{\eta}_{\theta_k}^{[m-1]} = \hat{\eta}_{\theta_k}^{[m-1]} + \mathsf{sl} \cdot h_{kj^*}(\cdot) \ ,$$

where sl is a small step-length ($0 < \mathsf{sl} \ll 1$). Therefore, only the best-performing base-learner (and therefore the best-performing covariate) contributes to the update.

(f) Set $\hat{\eta}_{\theta_k}^{[m]} = \hat{\eta}_{\theta_k}^{[m-1]}$.

(g) Repeat steps 4(b) to 4(f) for $k = 2, ..., 4$.

5 Iterate steps 3 and 4 until $m \geq m_{\mathsf{stop},k}$ for all $k = 1, ..., 4$.

Outer loop of gamboostLSS

Step $m + 1$

$$
\begin{aligned}
(\hat{\mu}^{[m]}, \hat{\sigma}^{[m]}, \hat{\nu}^{[m]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \hat{\eta}_{\mu}^{[\mathbf{m+1}]} \Longrightarrow \hat{\mu}^{[\mathbf{m+1}]} \,, \\
(\hat{\mu}^{[\mathbf{m+1}]}, \hat{\sigma}^{[m]}, \hat{\nu}^{[m]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \hat{\eta}_{\sigma}^{[\mathbf{m+1}]} \Longrightarrow \hat{\sigma}^{[\mathbf{m+1}]} \,, \\
(\hat{\mu}^{[\mathbf{m+1}]}, \hat{\sigma}^{[\mathbf{m+1}]}, \hat{\nu}^{[m]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \hat{\eta}_{\nu}^{[\mathbf{m+1}]} \Longrightarrow \hat{\nu}^{[\mathbf{m+1}]} \,, \\
(\hat{\mu}^{[\mathbf{m+1}]}, \hat{\sigma}^{[\mathbf{m+1}]}, \hat{\nu}^{[\mathbf{m+1}]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \hat{\eta}_{\tau}^{[\mathbf{m+1}]} \Longrightarrow \hat{\tau}^{[\mathbf{m+1}]} \,.
\end{aligned}
$$

Inner loop of gamboostLSS

Step $m+1, k=2, \triangleright\ \eta_{\theta_k} = \eta_\sigma$

- negative gradient vector:

$$\boldsymbol{u}_\sigma^{[m]} = \left(-\frac{\partial}{\partial \eta_\sigma} \rho(y_i, \eta_{\mu_i}^{[m+1]}, \eta_{\sigma_i}^{[m]}, \eta_{\nu_i}^{[m]}, \eta_{\tau_i}^{[m]}) \right)_{i=1,\ldots,n}$$

- component-wise fitting

$$\begin{aligned}
\boldsymbol{u}_\sigma^{[m]} &\xrightarrow{\text{base-learner}} \hat{h}_1(\boldsymbol{x}_1) \\
\boldsymbol{u}_\sigma^{[m]} &\xrightarrow{\text{base-learner}} \hat{h}_2(\boldsymbol{x}_2) \\
&\ldots \qquad\qquad \ldots \\
\boldsymbol{u}_\sigma^{[m]} &\xrightarrow{\text{base-learner}} \hat{h}_j^*(\boldsymbol{x}_{j^*}) \quad \text{best performer} \longrightarrow \text{included in } \eta_\sigma^{[m+1]} \\
&\ldots \qquad\qquad \ldots \\
\boldsymbol{u}_\sigma^{[m]} &\xrightarrow{\text{base-learner}} \hat{h}_J(\boldsymbol{x}_J)
\end{aligned}$$

Variable selection and shrinkage

- The main tuning parameter are the stopping iterations $m_{\text{stop},k}$. They control **variable selection** and the **amount of shrinkage**.
  - If boosting is stopped before convergence only the most important variables are included in the final model.
  - Variables that have never been selected in the updated step, are excluded.
  - Due to the small increments added in the update step, boosting incorporates shrinkage of effect sizes (compare to LASSO), leading to more stable predictions.
- For large $m_{\text{stop},k}$ boosting converges to the same solution as the original algorithm (in low-dimensional settings).
- The selection of $m_{\text{stop},k}$ is normally based on resampling methods, optimizing the predictive risk.

Example: Munich rental guide

- Provide precise point predictions and prediction intervals for the net-rent of flats in the city of Munich.
- Covariates: 238 categorical, 2 continuous and 1 spatial
- Observations: 3016 flats

Problem:

- Heteroscedasticity found in the data

Idea

Model not only the expected mean but also the variance $\Rightarrow$ **GAMLSS**

Munich rental guide

To deal with heteroscedasticity, we chose a three-parametric t-distribution with

$$\mathbb{E}(Y) = \mu \quad \text{and} \quad \text{Var}(Y) = \sigma^2 \frac{\texttt{df}}{\texttt{df} - 2}$$

For each of the parameters μ, σ, and df, we consider the predictors

$$\begin{aligned}
\eta_{\mu_i} &= \beta_{0\mu} + x_i^\top \beta_\mu + f_{1,\mu}(\text{size}_i) + f_{2,\mu}(\text{year}_i) + f_{\text{spat},\mu}(s_i) \ , \\
\eta_{\sigma_i} &= \beta_{0\sigma} + x_i^\top \beta_\sigma + f_{1,\sigma}(\text{size}_i) + f_{2,\sigma}(\text{year}_i) + f_{\text{spat},\sigma}(s_i) \ , \\
\eta_{\text{df}_i} &= \beta_{0\texttt{df}} + x_i^\top \beta_{\text{df}} + f_{1,\text{df}}(\text{size}_i) + f_{2,\text{df}}(\text{year}_i) + f_{\text{spat},\text{df}}(s_i) \ .
\end{aligned}$$

Base-learners

- Categorical variables: Simple linear models
- Continuous variables: P-splines
- Spatial variable: Gaussian Markov random field

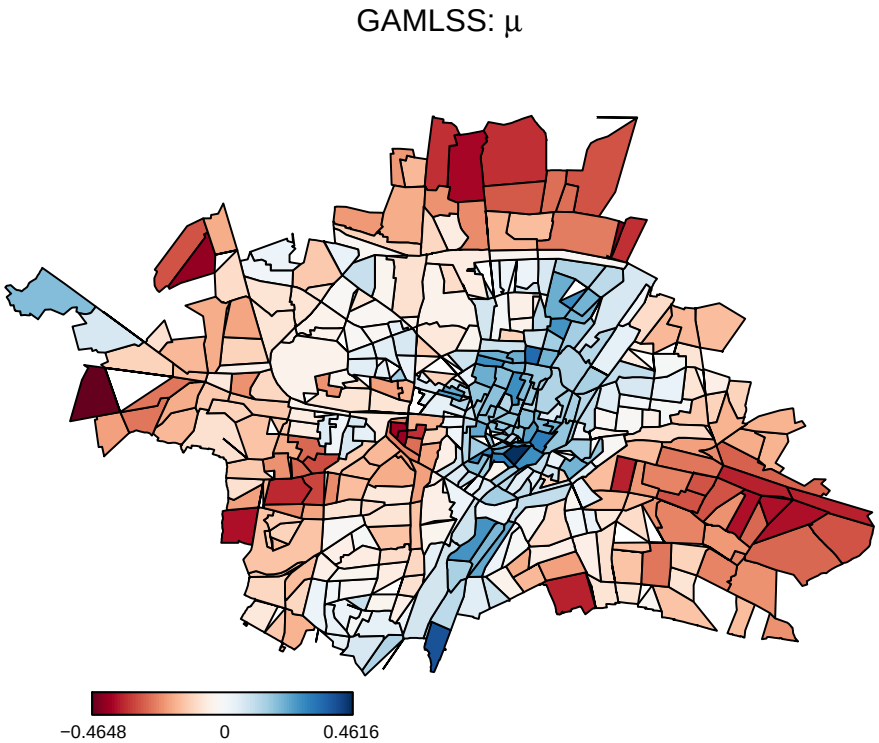
Munich rental guide

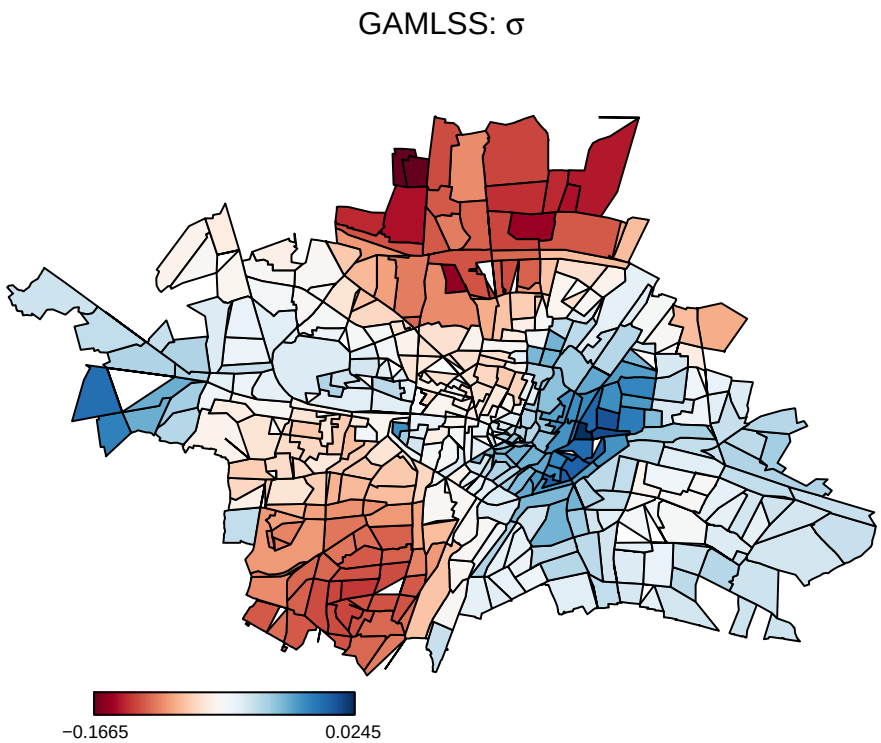
```
> library(gamboostLSS)
> form <- paste(names(data)[1], " ~ ",
                paste(names(data)[-c(1, 327, 328, 329)], collapse = " + "),
                " + bbs(wfl) + bbs(bamet) + bmrf(region, bnd = bound)")
> form <- as.formula(form)
> form
nmqms ~ erstbezg + dienstwg + gebmeist + gebgruen + hzkohojn +
    ... +
    bbs(wfl) + bbs(bamet) + bmrf(region, bnd = bound)
> ## Fit the model with (initially) 100 boosting steps
> mod <- gamboostLSS(formula = form, families = StudentTLSS(),
                     control = boost_control(mstop = 100,
                                             trace = TRUE),
                     baselearner = bols,
                     data = data)
[   1] ..................................... -- risk: 3294.323
[  41] ..................................... -- risk: 3091.206
[  81] ..................
Final risk: 3038.919
```

Munich rental guide

```
> ## optimal number of boosting iterations found by 3-dimensional
> ## cross-validation on a logarithmic grid resulted in
> ## 750 (mu), 108 (sigma), 235 (df) steps;
> ## Let model run until these values:
> mod[c(750, 108, 235)]
>
> ## Let's look at the number of variables per parameter:
> sel <- selected(mod)
> lapply(sel, function(x) length(unique(x)))

$mu
[1] 115

$sigma
[1] 31

$df
[1] 7

> ## (Very) sparse model (only 115, 31 and 5 base-learners out of 328)
```

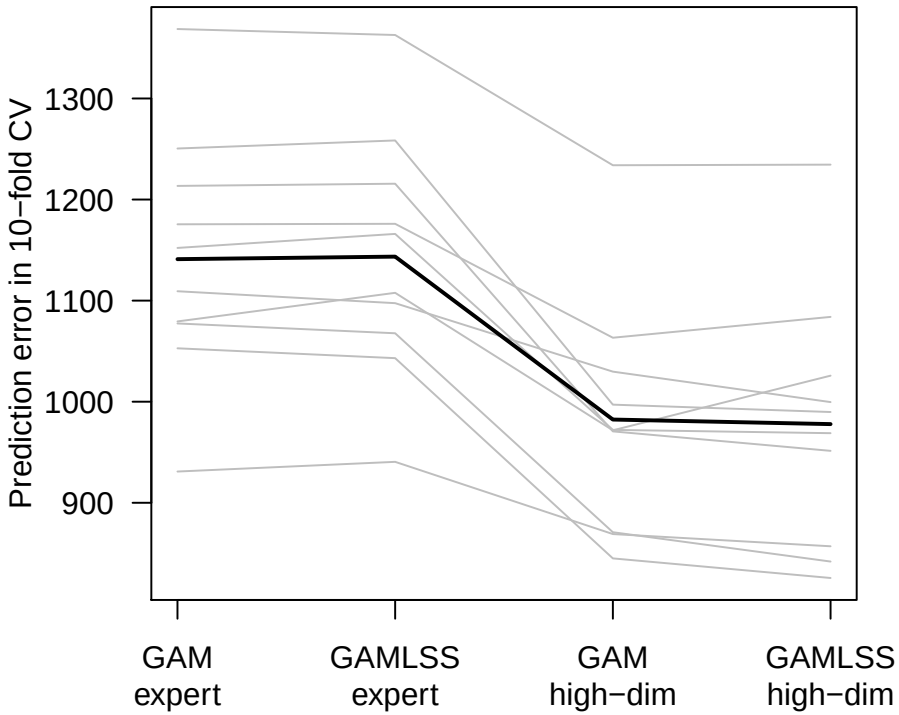
Results: spatial effects





Estimated spatial effects obtained for the high-dimensional GAMLSS for distribution parameters μ and σ. For the third parameter `df`, the corresponding variable was not selected.

Results: prediction error



Average mean squared prediction errors. Expert models contain only the 28 variables used in the official guide.

Example: Prediction of birth weight – gamboostLSS

- **Aim** Provide precise point predictions and prediction intervals for the birth weight of babies based on the last sonographic measurement before delivery

In this case, the predicted birth weight is 3710g ± 542g.

Example: Prediction of birth weight – gamboostLSS

- **Observations:**
  - 8712 babies delivered at the University Hospital of Erlangen
  - Various extreme cases with birth weight < 1500g or > 4500g
  - Size of prediction intervals should be child-specific
- **Predictors:**
  - 7 anthropometric measurements (head circumference, femur length, etc.)
  - Highly correlated (multicollinearity)
  - Additionally, interactions might be necessary

Solution:

Model not only the expected value, but also the variance with GAMLSS.
Automated variable selection via boosting $\Rightarrow$ `gamboostLSS`.

Example: Prediction of birth weight – gamboostLSS

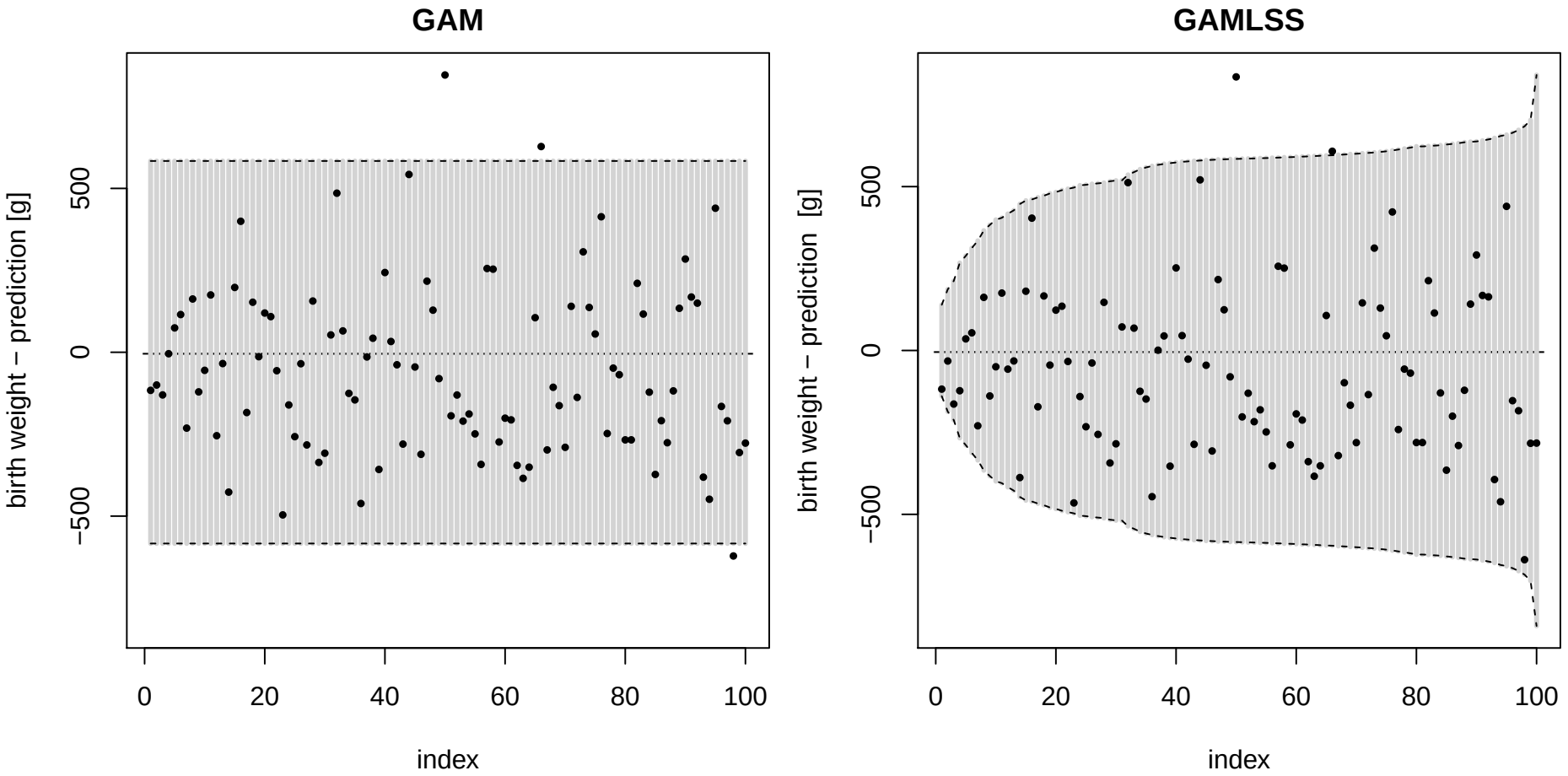
```
> library(gamboostLSS)
> form  ## Contains 29 base-learners

bw ~ bols(INT, intercept = FALSE) + bbs(BIPc) + bbs(FODc) +
     bbs(KUc) + bbs(ATDc) + bbs(ASDc) + bbs(AUc) + bbs(FLc) +
....
    bbs(ATDASD)

> ## Fit the model with (initially) 250 boosting steps
> mod <- gamboostLSS(formula = form, families = GaussianLSS(),
+                    control = boost_control(mstop = 250),
+                    data = dat.train)
> ## Takes less than 1 minute on standard laptop
> ## Tuning via mstop:
> cvr <- cvrisk(mod) ## Included in RForge version
> ## 25-fold bootstrap, 10x10 log-grid -> 2500 models
> ## incoporates parallel computing
> mod <- mod[mstop(cvr)]  ## mu = 73, sigma = 73
```

Example: Prediction of birth weight – gamboostLSS

```
> ##  compute the predictions
> pred.mu <- predict(mod, parameter = "mu", newdata = dat.test)
> pred.si <- predict(mod, parameter = "sigma", newdata = dat.test,
+                    type = "response")
> ## build the intervals
> lower.gamlss <- qnorm(p=0.025, mean = pred.mu, sd = pred.si)
> upper.gamlss <- qnorm(p=0.975, mean = pred.mu, sd = pred.si)
> summary((upper.gamlss - lower.gamlss)/2)## +/-

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  135.2  577.8   589.0   584.0   603.0   944.5

> ## Compare to GAM - fitted before by mboost
> summary((upper.gam - lower.gam)/2)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  585.5   585.5   585.5   585.5   585.5   585.5
```

Example: Prediction of birth weight

- **Result:** Prediction intervals by GAMLSS can express higher or lower uncertainty of the point prediction as also σ is baby-specific.



Summary

GAMLSS is a very flexible model class, extending the classical GAM framework. Not only the mean, but all distribution parameters are regressed to the predictors.

- The `gamlss` package is the user-friendly **gold standard** for low-dimensional settings
- `gamboostLSS` uses boosting to fit the different sub-models:
  - Works in **high-dimensional** settings
  - Incorporates **variable selection**
  - Can include various type of predictor effects (linear, smooth, spatial, etc.)
  - Will be further extended ...
  - The initial package version is on CRAN, new features on R-Forge.

References

- Rigby, R. A. and Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Applied Statistics*, **54**, 507-554.
- Mayr, A., Fenske, N., Hofner,B., Kneib, T. and Schmid, M. (2012). Generalized additive models for location, scale and shape for high-dimensional data – a flexible approach based on boosting. *Applied Statistics*, **61**(3), 403-427.
- Hofner, B., Mayr, A., Fenske, N. and Schmid, M. (2013). gamboostLSS: Boosting Methods for GAMLSS Models, R package version 1.1-0.
- Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M., Hofner, B, Sobotka, F. and Scheipl, F. (2013). mboost: Model-based Boosting, R package version 2.2-2.
- Stasinopoulos, D. M. and Rigby, R.A. (2013). gamlss: Generalized Additive Models for Location Scale and Shape, R package version 4.2-0.