Boosting functional regression models with `FDboost`

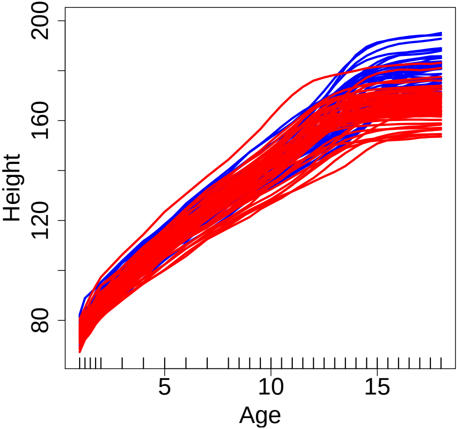
Sarah Brockhaus & David Rügamer

In collaboration with Sonja Greven, Thorsten Hothorn, Andy Mayr, Fabian Scheipl and Almond Stöcker

LMU Munich

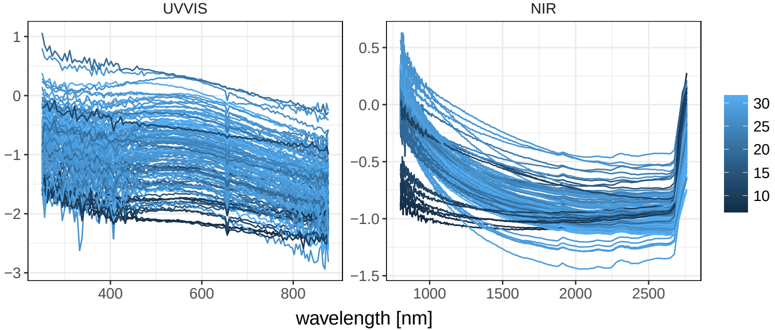
July 24, 2017

Functional data: Growth curves



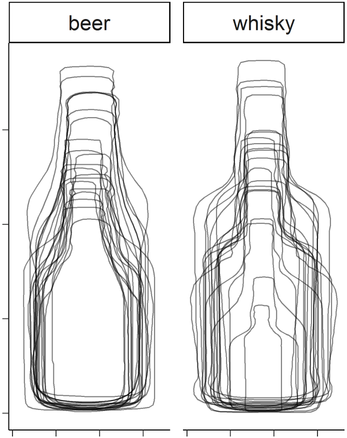
(Ramsay and Silverman, 2005)

Functional data: Spectrometric measures



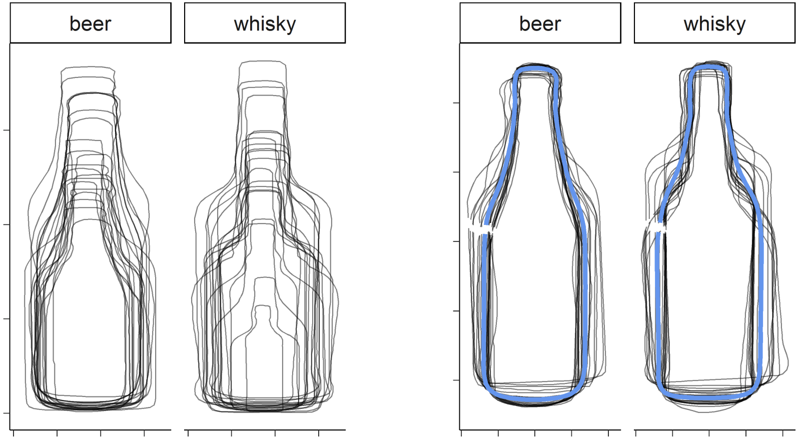
(Fuchs et al., 2015)

Functional data: Shapes



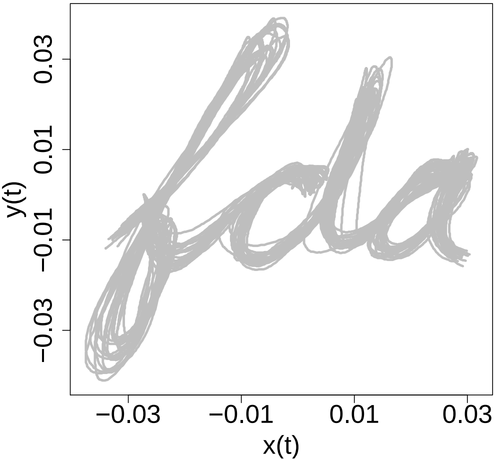
(Bonhomme et al., 2014)

Functional data: Shapes



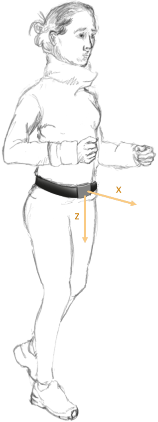
(Bonhomme et al., 2014)

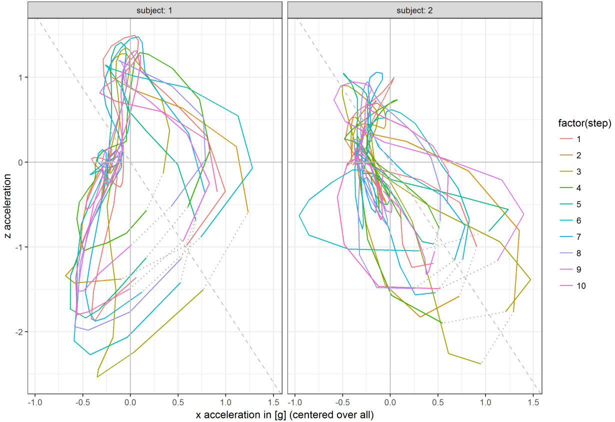
Functional data: Trajectories



(Ramsay and Silverman, 2005)

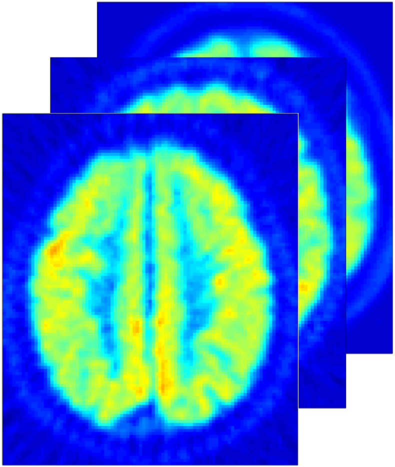
Functional data: Movement



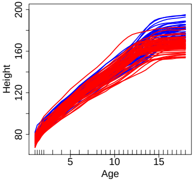


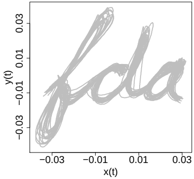
(Martin Daumer, Sylvia Lawry Centre for Multiple Sclerosis Research)

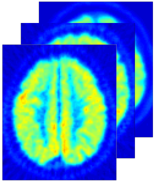
Functional data: Brain scans



Introduction to functional data

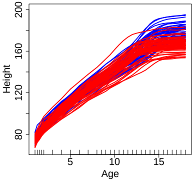


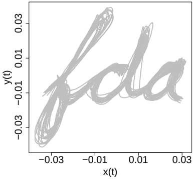


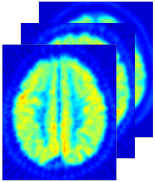


- Observation units are functions; several measurement points for each observation unit

Introduction to functional data

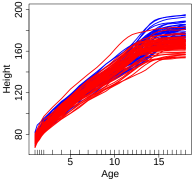


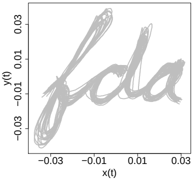


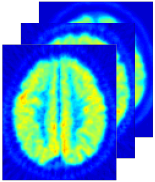


- Observation units are functions; several measurement points for each observation unit
- Measurement points on regular or irregular grid

Introduction to functional data

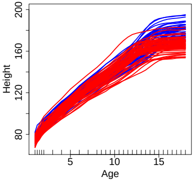


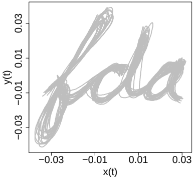


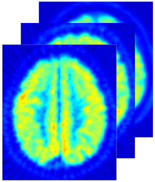


- Observation units are functions; several measurement points for each observation unit
- Measurement points on regular or irregular grid
- Possibly arbitrary many measurements possible
  $\rightarrow$ smooth data generating function

Introduction to functional data

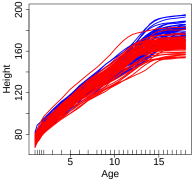


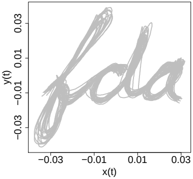


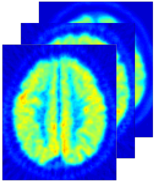


- Observation units are functions; several measurement points for each observation unit
- Measurement points on regular or irregular grid
- Possibly arbitrary many measurements possible
  $\rightarrow$ smooth data generating function
- Observations possibly with (measurement) error

Introduction to functional data







- Observation units are functions; several measurement points for each observation unit
- Measurement points on regular or irregular grid
- Possibly arbitrary many measurements possible
  $\rightarrow$ smooth data generating function
- Observations possibly with (measurement) error
- Difference: functional data $\leftrightarrow$ time series

Outline

Basic statistics for functional data

Mean, Variance and Covariance

- functional variable $X(t)$,
  with $t \in \mathcal{T}$ and $\mathcal{T}$ interval in $\mathbb{R}$
- sample $x_i(t)$, $i = 1, \ldots, n$

Mean, Variance and Covariance

- functional variable $X(t)$,
  with $t \in \mathcal{T}$ and $\mathcal{T}$ interval in $\mathbb{R}$
- sample $x_i(t)$, $i = 1, \ldots, n$
- functional mean:

$$\hat{\mu}_X(t) = \bar{x}(t) = \frac{1}{n} \sum_{i=1}^{n} x_i(t)$$

- functional variance:

$$\hat{\sigma}_X(t) = \frac{1}{n-1} \sum_{i=1}^{n} [x_i(t) - \bar{x}(t)]^2$$

Mean, Variance and Covariance

- functional variable $X(t)$, with $t \in \mathcal{T}$ and $\mathcal{T}$ interval in $\mathbb{R}$
- sample $x_i(t)$, $i = 1, \ldots, n$
- functional mean:

$$\hat{\mu}_X(t) = \bar{x}(t) = \frac{1}{n} \sum_{i=1}^{n} x_i(t)$$

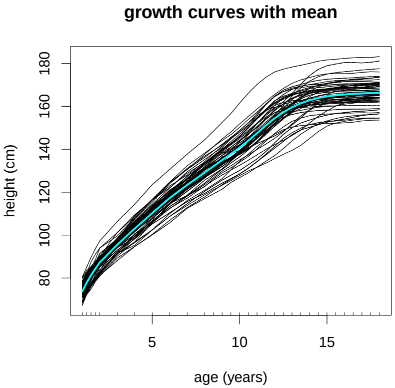
- functional variance:

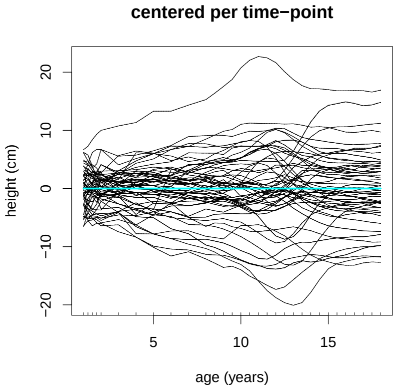
$$\hat{\sigma}_X(t) = \frac{1}{n-1} \sum_{i=1}^{n} [x_i(t) - \bar{x}(t)]^2$$

- functional (auto-)covariance:

$$\hat{\sigma}_X(t_1, t_2) = \frac{1}{n-1} \sum_{i=1}^{n} [x_i(t_1) - \bar{x}(t_1)][x_i(t_2) - \bar{x}(t_2)]$$

Example for mean: Growth curves of 54 girls





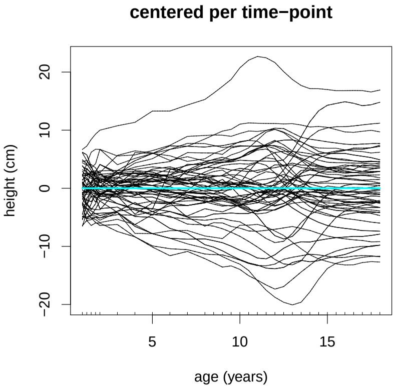
estimated mean:

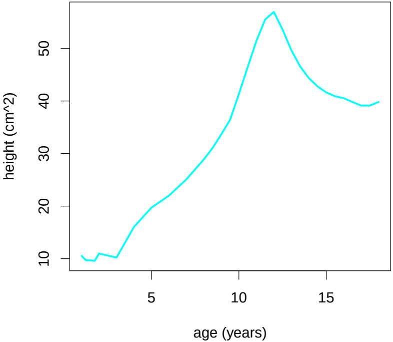
$$\hat{\mu}_X(t) = \bar{x}(t) = \frac{1}{n}\sum_{i=1}^{n} x_i(t)$$

centered curves:

$$x_i^*(t) = x_i(t) - \bar{x}(t)$$

Example for variance





centered curves:

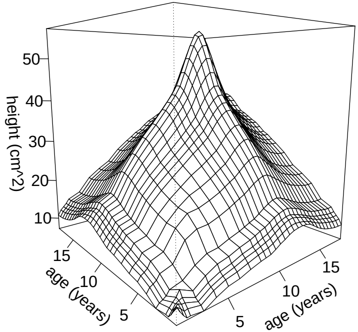
$$x_i^*(t) = x_i(t) - \bar{x}(t)$$

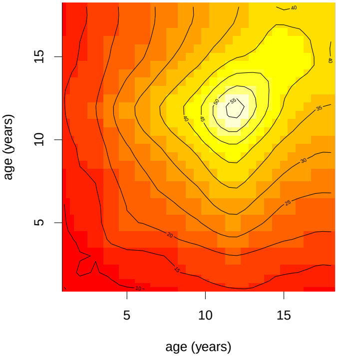
estimated variance:

$$\hat{\sigma}_X(t) = \frac{1}{n-1}\sum_{i=1}^{n}[x_i(t) - \bar{x}(t)]^2$$

Example for covariance surface

$$\hat{\sigma}_X(t_1, t_2) = \frac{1}{n-1}\sum_{i=1}^{n}[x_i(t_1) - \bar{x}(t_1)][x_i(t_2) - \bar{x}(t_2)]$$



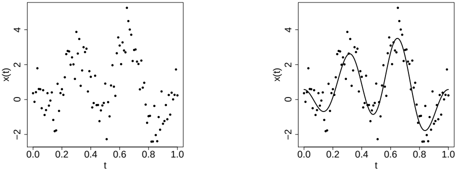


Functional data analysis in a nutshell

Outlook to functional data analysis

Important topics: (Ramsay and Silverman, 2005)

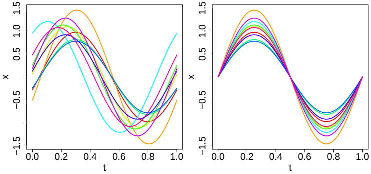
- Data representation → interpolation, smoothing

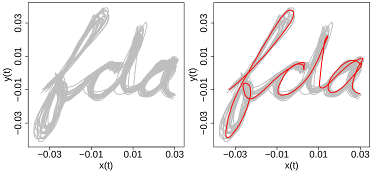


Outlook to functional data analysis

Important topics: (Ramsay and Silverman, 2005)

- Data representation $\rightarrow$ interpolation, smoothing
- Visualization $\rightarrow$ registration, outlier detection

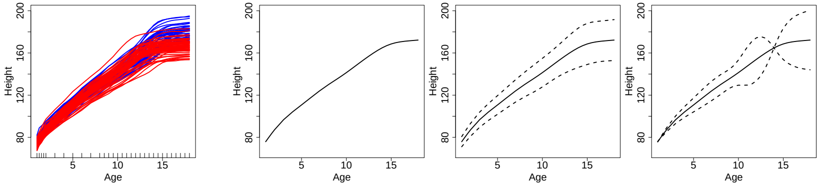




Outlook to functional data analysis

Important topics: (Ramsay and Silverman, 2005)

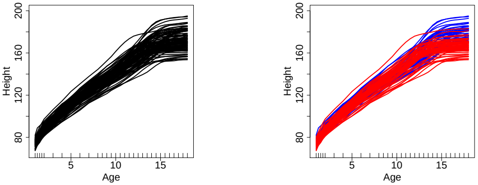
- Data representation $\rightarrow$ interpolation, smoothing
- Visualization $\rightarrow$ registration, outlier detection
- Finding of patterns in the variation of the data $\rightarrow$ functional principal component analysis (FPCA)



Outlook to functional data analysis

Important topics: (Ramsay and Silverman, 2005)

- Data representation $\rightarrow$ interpolation, smoothing
- Visualization $\rightarrow$ registration, outlier detection
- Finding of patterns in the variation of the data $\rightarrow$ functional principal component analysis (FPCA)
- Classification and clustering



Outlook to functional data analysis

Important topics: (Ramsay and Silverman, 2005)

- Data representation $\rightarrow$ interpolation, smoothing
- Visualization $\rightarrow$ registration, outlier detection
- Finding of patterns in the variation of the data $\rightarrow$ functional principal component analysis (FPCA)
- Classification and clustering
- Regression $\rightarrow$ functional regression models (Morris, 2015; Greven and Scheipl, 2017)

scalar-on-function: $$y_i = \beta_0 + \int x_i(s)\beta(s)\,ds + \varepsilon_i$$

function-on-scalar: $$y_i(t) = \beta_0(t) + x_i\beta(t) + \varepsilon_i(t)$$

function-on-function: $$y_i(t) = \beta_0(t) + \int x_i(s)\beta(s,t)\,ds + \varepsilon_i(t)$$

R packages

Visualization

- Shang & Hyndman (2016). *rainbow: Rainbow Plots, Bagplots and Boxplots for Functional Data.* R package version 3.4. https://CRAN.R-project.org/package=rainbow

Visualization, descriptive and exploratory analysis

- Febrero-Bande & Oviedo de la Fuente (2012). *Statistical Computing in Functional Data Analysis: The R Package fda.usc.* Journal of Statistical Software, 51(4), 1–28.
- Ramsay, Wickham, Graves & Hooker (2014). *fda: Functional Data Analysis.* R package version 2.4.4. https://CRAN.R-project.org/package=fda

Regression

- Goldsmith, Scheipl, Huang, Wrobel, Gellar, Harezlak, McLean, Swihart, Xiao, Crainiceanu & Reiss (2016). *refund: Regression with Functional Data.* R package version 0.1-16. https://CRAN.R-project.org/package=refund
- Brockhaus & Rügamer (2017). *FDboost: Boosting Functional Regression models.* R package version 0.3-0. https://CRAN.R-project.org/package=FDboost

see also the CRAN Task View: Functional Data Analysis

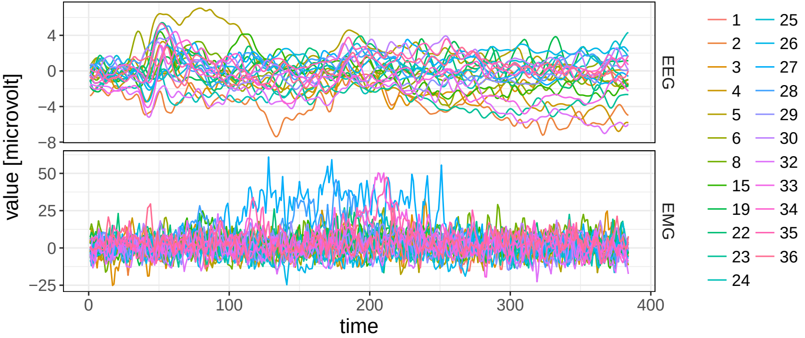
Regression with functional data

Motivation: emotion components data (I)

Data set from Gentsch et al. (2014), also used in Rügamer et al. (2016)

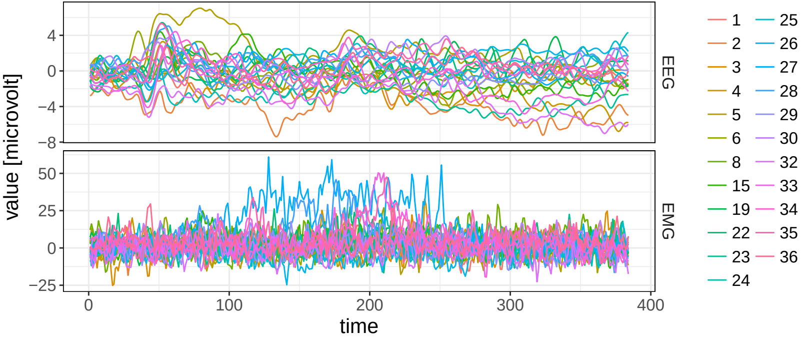
- Main goal: Understand how emotions evolve
- Participants played a gambling game with real money outcome
- Emotions “measured” via EMG (muscle activity in the face)
- Influencing factor *appraisals* measured via EEG (brain activity)
- Different game situation, a lot of trials

Motivation: emotion components data (II)



(Brockhaus et al., 2017)

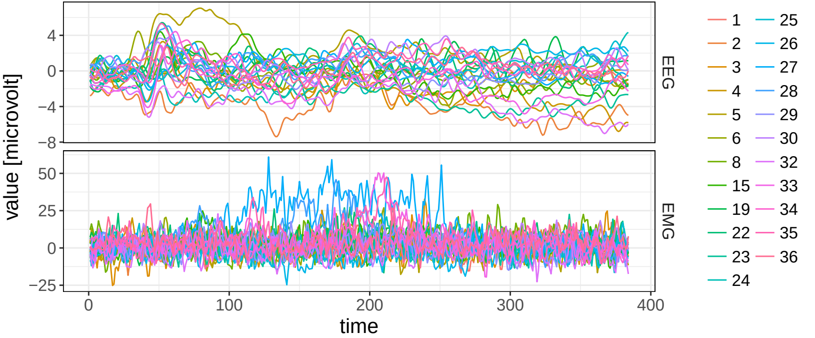
Motivation: emotion components data (II)



(Brockhaus et al., 2017)

Function-on-function-regression

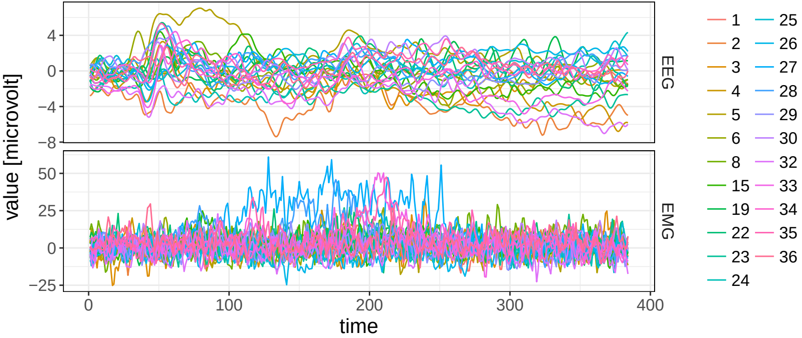
Motivation: emotion components data (II)



(Brockhaus et al., 2017)

Function-on-function-regression . . . what for?

Motivation: emotion components data (II)

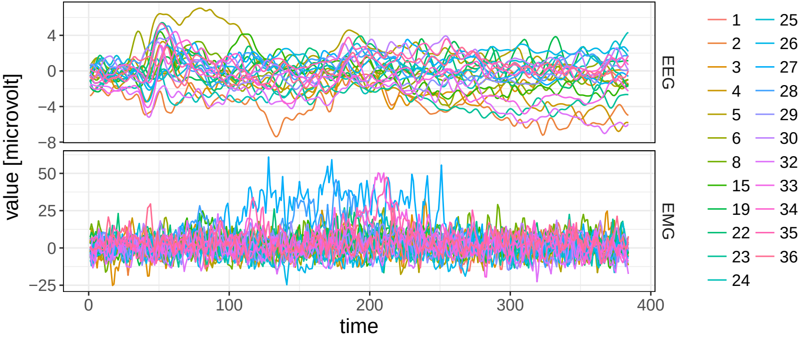


(Brockhaus et al., 2017)

Function-on-function-regression . . . what for?

- The absolute value is not really of interest

Motivation: emotion components data (II)



(Brockhaus et al., 2017)

Function-on-function-regression . . . what for?

- The absolute value is not really of interest
- Describe the course of each curve, more specifically their relationship → average course of function

Generic additive regression model

- functional response $Y(t)$, $t \in \mathcal{T} = [T_1, T_2]$
- vector of covariates $\boldsymbol{x}$ containing functional covariates $x(s)$ and scalar covariates z

Generic model

$$\mathbb{E}\left(Y(t) \mid \boldsymbol{x}\right) = h(\boldsymbol{x}, t) = \sum\nolimits_j h_j(\boldsymbol{x}, t)$$

- $h(\boldsymbol{x}, t)$ linear predictor which is the sum of partial effects $h_j(\boldsymbol{x}, t)$
- each $h_j(\boldsymbol{x}, t)$ is a real valued function over $\mathcal{T}$ and can depend on or several covariates

Generic additive regression model

- functional response $Y(t)$, $t \in \mathcal{T} = [T_1, T_2]$
- vector of covariates $\boldsymbol{x}$ containing functional covariates $x(s)$ and scalar covariates z

Generic model

$$\mathbb{E}\left(Y(t) \mid \boldsymbol{x}\right) = h(\boldsymbol{x}, t) = \sum\nolimits_j h_j(\boldsymbol{x}, t)$$

- $h(\boldsymbol{x}, t)$ linear predictor which is the sum of partial effects $h_j(\boldsymbol{x}, t)$
- each $h_j(\boldsymbol{x}, t)$ is a real valued function over $\mathcal{T}$ and can depend on or several covariates

$\rightarrow$ Scalar response as degenerated case with $\mathcal{T} = [T_1, T_1]$

Generic additive regression model

- functional response $Y(t)$, $t \in \mathcal{T} = [T_1, T_2]$
- vector of covariates $\boldsymbol{x}$ containing functional covariates $x(s)$ and scalar covariates z

Generic model

$$\mathbb{E}\left(Y(t) \mid \boldsymbol{x}\right) = h(\boldsymbol{x}, t) = \sum\nolimits_j h_j(\boldsymbol{x}, t)$$

- $h(\boldsymbol{x}, t)$ linear predictor which is the sum of partial effects $h_j(\boldsymbol{x}, t)$
- each $h_j(\boldsymbol{x}, t)$ is a real valued function over $\mathcal{T}$ and can depend on or several covariates

$\rightarrow$ Scalar response as degenerated case with $\mathcal{T} = [T_1, T_1]$

Partial effects $h_j(x, t)$ of scalar covariates

- smooth intercept $\beta_0(t)$
- group-specific smooth intercepts $\beta_{0a}(t)$

- smooth linear effect of scalar covariate $z\beta(t)$
- smooth non-linear effect of scalar covariate $g(z, t)$

- interactions, e.g., $z_1 z_2 \beta(t)$ and $g(z_1, z_2, t)$

(Scheipl et al., 2015; Brockhaus et al., 2015)

Partial effects $h_j(x, t)$ of functional covariates

- concurrent effect $x(t)\beta(t)$

Partial effects $h_j(x, t)$ of functional covariates

- concurrent effect $x(t)\beta(t)$
- linear effect of functional covariate $\int_{\mathcal{S}} x(s)\beta(s,t)\, ds$

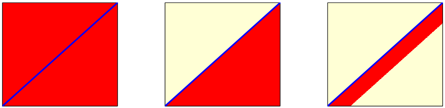
Partial effects $h_j(x, t)$ of functional covariates

- concurrent effect $x(t)\beta(t)$
- linear effect of functional covariate $\int_{\mathcal{S}} x(s)\beta(s, t)\, ds$
- constrained effect of functional covariate $\int_{l(t)}^{u(t)} x(s)\beta(s, t)\, ds$

Partial effects $h_j(x, t)$ of functional covariates

- concurrent effect $x(t)\beta(t)$
- linear effect of functional covariate $\int_{\mathcal{S}} x(s)\beta(s, t)\, ds$
- constrained effect of functional covariate $\int_{l(t)}^{u(t)} x(s)\beta(s, t)\, ds$

effect	linear	historical	lag
$[l(t), u(t)]$	$[T_1, T_2]$	$[T_1, t]$	$[t-\delta, t]$

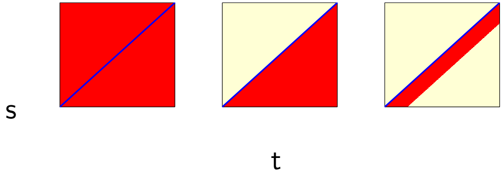


(Scheipl et al., 2015; Brockhaus et al., 2016b,a)

Partial effects $h_j(x, t)$ of functional covariates

- concurrent effect $x(t)\beta(t)$
- linear effect of functional covariate $\int_{\mathcal{S}} x(s)\beta(s, t)\, ds$
- constrained effect of functional covariate $\int_{l(t)}^{u(t)} x(s)\beta(s, t)\, ds$

effect	linear	historical	lag
$[l(t), u(t)]$	$[T_1, T_2]$	$[T_1, t]$	$[t - \delta, t]$



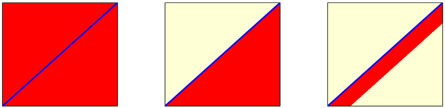
(Scheipl et al., 2015; Brockhaus et al., 2016b,a)

Partial effects $h_j(x, t)$ of functional covariates

- concurrent effect $x(t)\beta(t)$
- linear effect of functional covariate $\int_{\mathcal{S}} x(s)\beta(s, t)\, ds$
- constrained effect of functional covariate $\int_{l(t)}^{u(t)} x(s)\beta(s, t)\, ds$

effect	linear	historical	lag
$[l(t), u(t)]$	$[T_1, T_2]$	$[T_1, t]$	$[t - \delta, t]$

s



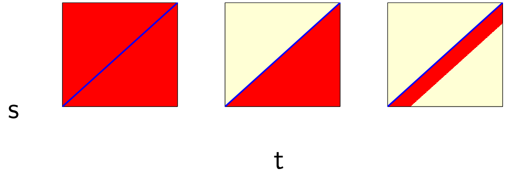
t

(Scheipl et al., 2015; Brockhaus et al., 2016b,a)

Partial effects $h_j(x, t)$ of functional covariates

- concurrent effect $x(t)\beta(t)$
- linear effect of functional covariate $\int_{\mathcal{S}} x(s)\beta(s, t)\, ds$
- constrained effect of functional covariate $\int_{l(t)}^{u(t)} x(s)\beta(s, t)\, ds$

effect	linear	historical	lag
$[l(t), u(t)]$	$[T_1, T_2]$	$[T_1, t]$	$[t - \delta, t]$



(Scheipl et al., 2015; Brockhaus et al., 2016b,a)

Interactions of functional and scalar covariates

- linear interaction of scalar and functional covariate

$$z \int_{l(t)}^{u(t)} x(s)\beta(s,t)ds$$

- group-specific functional effects

$$I(z = a) \cdot \int_{l(t)}^{u(t)} x(s)\beta_a(s,t)ds$$

with indicator function $I(\cdot)$

Interactions of functional and scalar covariates

- linear interaction of scalar and functional covariate

$$z \int_{l(t)}^{u(t)} x(s)\beta(s,t)ds$$

- group-specific functional effects

$$I(z = a) \cdot \int_{l(t)}^{u(t)} x(s)\beta_a(s,t)ds$$

with indicator function $I(\cdot)$

$\rightarrow$ For all the listed effects ensure identifiability by suitable constraints

Specification of partial effects

The generic model: $\mathbb{E}(Y(t)|\boldsymbol{x}) = h(\boldsymbol{x}, t) = \sum_j h_j(\boldsymbol{x}, t)$

Row tensor product basis

$$h_j(\boldsymbol{x}, t) = \left\{ \boldsymbol{b}_j(\boldsymbol{x}, t)^\top \odot \boldsymbol{b}_Y(t)^\top \right\} \boldsymbol{\theta}_j$$

- $\boldsymbol{b}_j$ / $\boldsymbol{b}_Y$ vector of κ_j/κ_Y basis functions in covariates / over $\mathcal{T}$
- $\odot$ row-wise tensor product ('Kronecker product on rows')
- $\boldsymbol{\theta}_j$ coefficient vector

- Ridge-type penalty with penalty term $\boldsymbol{\theta}_j^\top \mathbf{P}_{jY} \boldsymbol{\theta}_j$ for regularization in both directions

Specification of partial effects

The generic model: $\mathbb{E}(Y(t)|\boldsymbol{x}) = h(\boldsymbol{x}, t) = \sum_j h_j(\boldsymbol{x}, t)$

Row tensor product basis

$$h_j(\boldsymbol{x}, t) = \left\{ \boldsymbol{b}_j(\boldsymbol{x}, t)^\top \odot \boldsymbol{b}_Y(t)^\top \right\} \boldsymbol{\theta}_j$$

- $\boldsymbol{b}_j$ / $\boldsymbol{b}_Y$ vector of κ_j/κ_Y basis functions in covariates / over $\mathcal{T}$
- $\odot$ row-wise tensor product ('Kronecker product on rows')
- $\boldsymbol{\theta}_j$ coefficient vector
- Ridge-type penalty with penalty term $\boldsymbol{\theta}_j^\top \mathbf{P}_{jY} \boldsymbol{\theta}_j$ for regularization in both directions
- if possible, representation as generalized linear array model (Currie et al., 2006).

Estimation

How do we estimate such models?

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

  - a factor-specific historical effect $\int_0^t x(s)\beta_a(s,t)\, ds$

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

  - a factor-specific historical effect $\int_0^t x(s)\beta_a(s,t)\,ds$
  - factor with $\kappa_z = 10$ levels

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

  - a factor-specific historical effect $\int_0^t x(s)\beta_a(s,t)\,ds$
  - factor with $\kappa_z = 10$ levels
  - $\kappa_s = \kappa_t = 20$ B-spline bases for $\beta_a(s,t)$ smoothness in s and t

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

  - a factor-specific historical effect $\int_0^t x(s)\beta_a(s,t)\,ds$
  - factor with $\kappa_z = 10$ levels
  - $\kappa_s = \kappa_t = 20$ B-spline bases for $\beta_a(s,t)$ smoothness in s and t

  $\Rightarrow \text{ncol}(\boldsymbol{X}) = \kappa_z \cdot \kappa_s \cdot \kappa_t$

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

  - a factor-specific historical effect $\int_0^t x(s)\beta_a(s,t)\,ds$
  - factor with $\kappa_z = 10$ levels
  - $\kappa_s = \kappa_t = 20$ B-spline bases for $\beta_a(s,t)$ smoothness in s and t

  $\Rightarrow \text{ncol}(\boldsymbol{X}) = \kappa_z \cdot \kappa_s \cdot \kappa_t = 10 \cdot 20 \cdot 20 = 4000$

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

  - a factor-specific historical effect $\int_0^t x(s)\beta_a(s,t)\,ds$
  - factor with $\kappa_z = 10$ levels
  - $\kappa_s = \kappa_t = 20$ B-spline bases for $\beta_a(s,t)$ smoothness in s and t

  $\Rightarrow \mathrm{ncol}(\boldsymbol{X}) = \kappa_z \cdot \kappa_s \cdot \kappa_t = 10 \cdot 20 \cdot 20 = 4000$
- So how can we handle and fit multiple of such partial effects at the same time?

Estimation

How do we estimate such models?

- As per usual: Write down the (penalized) log-likelihood and maximize it (see, e.g., Scheipl et al., 2016)
- Problem: Computational feasibility

  $\rightarrow$ For example, consider a model with

  - a factor-specific historical effect $\int_0^t x(s)\beta_a(s,t)\,ds$
  - factor with $\kappa_z = 10$ levels
  - $\kappa_s = \kappa_t = 20$ B-spline bases for $\beta_a(s,t)$ smoothness in s and t

  $\Rightarrow \text{ncol}(\boldsymbol{X}) = \kappa_z \cdot \kappa_s \cdot \kappa_t = 10 \cdot 20 \cdot 20 = 4000$

- So how can we handle and fit multiple of such partial effects at the same time?

$\rightarrow$ component-wise boosting

Estimation by component-wise gradient boosting

Idea:

Estimation by component-wise gradient boosting

Idea:

- iteratively boost the model performance ($\hat{=}$ reduce expected L_2-loss)

Estimation by component-wise gradient boosting

Idea:

- iteratively boost the model performance
  ($\hat{=}$ reduce expected L_2-loss)
- by fitting and evaluating the partial effects component-wise

Estimation by component-wise gradient boosting

Idea:

- iteratively boost the model performance
  ($\hat{=}$ reduce expected L_2-loss)
- by fitting and evaluating the partial effects component-wise
- using one partial effect at a time to update the model

Estimation by component-wise gradient boosting

Idea:

- iteratively boost the model performance
  ($\hat{=}$ reduce expected L_2-loss)
- by fitting and evaluating the partial effects component-wise
- using one partial effect at a time to update the model

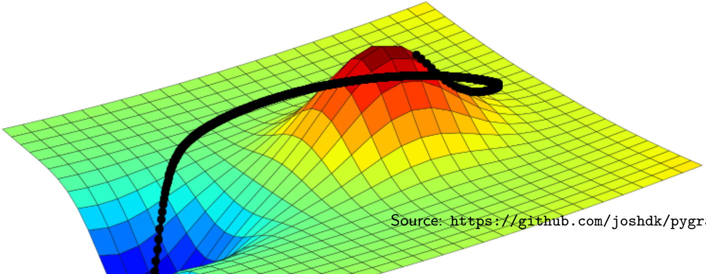
$\rightarrow$ Results in component-wise gradient descent steps

Estimation by component-wise gradient boosting

Idea:

- iteratively boost the model performance ($\hat{=}$ reduce expected L_2-loss)
- by fitting and evaluating the partial effects component-wise
- using one partial effect at a time to update the model

$\rightarrow$ Results in component-wise gradient descent steps



Source: https://github.com/joshdk/pygradesc

Component-wise gradient boosting: algorithm

Goal of boosting: Minimize the expected loss
Use the (penalized) regression models for effects h_j as base-learners

Algorithm For boosting iterations $m = 1, \ldots, m_{\text{stop}}$:

- compute the negative gradient $u_i(t)$ of the expected loss using the current estimate of the linear predictor $\hat{h}^{[m]}(x_i, t)$
- fit each base-learner to $u_i(t)$
- select the best fitting base-learner
- update the according parameters using a fixed step-length $\nu \in (0, 1]$

The final model is a linear combination of base-learner fits.

(Brockhaus et al., 2015)

Remember the generic model

- functional response $Y(t)$, $t \in \mathcal{T} \subset \mathbb{R}$
- vector of covariates $\boldsymbol{x}$ containing functional covariates $x(s)$ and scalar covariates z

Generic model

$$\mathbb{E}\left(Y(t) \mid \boldsymbol{x}\right) = h(\boldsymbol{x}, t) = \sum\nolimits_j h_j(\boldsymbol{x}, t)$$

- $h(\boldsymbol{x}, t)$ linear predictor which is the sum of partial effects $h_j(\boldsymbol{x}, t)$
- each $h_j(\boldsymbol{x}, t)$ is a real valued function over $\mathcal{T}$ and can depend on or several covariates

Remember the generic model

- functional response $Y(t)$, $t \in \mathcal{T} \subset \mathbb{R}$
- vector of covariates $\boldsymbol{x}$ containing functional covariates $x(s)$ and scalar covariates z

Generic model

$$\xi\left(Y(t) \mid \boldsymbol{x}\right) = h(\boldsymbol{x}, t) = \sum\nolimits_j h_j(\boldsymbol{x}, t)$$

- $h(\boldsymbol{x}, t)$ linear predictor which is the sum of partial effects $h_j(\boldsymbol{x}, t)$
- each $h_j(\boldsymbol{x}, t)$ is a real valued function over $\mathcal{T}$ and can depend on or several covariates

Remember the generic model

- functional response $Y(t)$, $t \in \mathcal{T} \subset \mathbb{R}$
- vector of covariates $\boldsymbol{x}$ containing functional covariates $x(s)$ and scalar covariates z

Generic model

$$\xi\left(Y(t) \mid \boldsymbol{x}\right) = h(\boldsymbol{x}, t) = \sum\nolimits_j h_j(\boldsymbol{x}, t)$$

- $h(\boldsymbol{x}, t)$ linear predictor which is the sum of partial effects $h_j(\boldsymbol{x}, t)$
- each $h_j(\boldsymbol{x}, t)$ is a real valued function over $\mathcal{T}$ and can depend on or several covariates

- ξ is some transformation function, e.g., a quantile or $\mathbb{E}$

Transformation and loss functions

The generic model: $\xi(Y(t)|\boldsymbol{x}) = h(\boldsymbol{x}, t) = \sum_j h_j(\boldsymbol{x}, t)$

Transformation and loss functions

The generic model: $\xi(Y(t)|\boldsymbol{x}) = h(\boldsymbol{x}, t) = \sum_j h_j(\boldsymbol{x}, t)$

model	ξ	loss function ρ
LM	$\mathbb{E}$	L_2-loss
GLM	$g \circ \mathbb{E}$	negative log-likelihood
median regression	$Q_{0.5}$	L_1-loss
quantile regression	Q_τ	check function

Transformation and loss functions

The generic model: $\xi(Y(t)|\boldsymbol{x}) = h(\boldsymbol{x}, t) = \sum_j h_j(\boldsymbol{x}, t)$

model	ξ	loss function ρ
LM	$\mathbb{E}$	L_2-loss
GLM	$g \circ \mathbb{E}$	negative log-likelihood
median regression	$Q_{0.5}$	L_1-loss
quantile regression	Q_τ	check function

Transformation and loss functions

The generic model: $\xi(Y(t)|\boldsymbol{x}) = h(\boldsymbol{x}, t) = \sum_j h_j(\boldsymbol{x}, t)$

model	ξ	loss function ρ
LM	$\mathbb{E}$	L_2-loss
GLM	$g \circ \mathbb{E}$	negative log-likelihood
median regression	$Q_{0.5}$	L_1-loss
quantile regression	Q_τ	check function

$\rightarrow$ GAMLSS (Rigby and Stasinopoulos, 2005) also possible

Transformation and loss functions

The generic model: $\xi(Y(t)|\boldsymbol{x}) = h(\boldsymbol{x}, t) = \sum_j h_j(\boldsymbol{x}, t)$

model	ξ	loss function ρ
LM	$\mathbb{E}$	L_2-loss
GLM	$g \circ \mathbb{E}$	negative log-likelihood
median regression	$Q_{0.5}$	L_1-loss
quantile regression	Q_τ	check function

- → GAMLSS (Rigby and Stasinopoulos, 2005) also possible
- → Loss function for trajectories $\hat{=}$ integrated loss over domain of response:

$$\ell(Y, h(\boldsymbol{x})) = \int_{\mathcal{T}} \underbrace{\rho(Y(t), h(\boldsymbol{x}, t))}_{\text{pointwise loss for } t} dt$$

Tuning, early stopping and model selection

- Tuning
  - We fix the ν and degrees of freedom for each baselearner
  - number of boosting iterations determines model complexity
  - optimal stopping iteration is determined by resampling methods (on the level of curves)

Tuning, early stopping and model selection

- Tuning
  - We fix the ν and degrees of freedom for each baselearner
  - number of boosting iterations determines model complexity
  - optimal stopping iteration is determined by resampling methods (on the level of curves)
- Early stopping
  - Regularization technique to avoid overfitting
  - Induces parameter shrinkage
  - and model selection

Tuning, early stopping and model selection

- Tuning
  - We fix the ν and degrees of freedom for each baselearner
  - number of boosting iterations determines model complexity
  - optimal stopping iteration is determined by resampling methods (on the level of curves)

- Early stopping
  - Regularization technique to avoid overfitting
  - Induces parameter shrinkage
  - and model selection

- Alternatively: Model selection via stability selection (Meinshausen and Bühlmann, 2010; Shah and Samworth, 2013)

Implementation

Implemented in

- R package `FDboost` (Brockhaus and Rügamer, 2017)
- based on R package `mboost` (Hothorn et al., 2017)
- Extension to GAMLSS based on `gamboostLSS` (Hofner et al., 2017)

hosted on `https://github.com/boost-R`

Implementation in `FDboost` (I)

Goal: Make use of the modular implementation of `mboost`

Implementation in FDboost (I)

Goal: Make use of the modular implementation of mboost

- Scalar-on-function regression:
  the loss and empirical risk is just as for 'scalar-on-scalar regression'

Implementation in `FDboost` (I)

Goal: Make use of the modular implementation of `mboost`

- Scalar-on-function regression:
  the loss and empirical risk is just as for 'scalar-on-scalar regression'
  - We just have to implement new base-learners and

Implementation in FDboost (I)

Goal: Make use of the modular implementation of mboost

- Scalar-on-function regression:
  the loss and empirical risk is just as for 'scalar-on-scalar regression'
  - We just have to implement new base-learners and
  - allow for different data input (e.g. matrices for curve observations)

Implementation in FDboost (I)

Goal: Make use of the modular implementation of `mboost`

- Scalar-on-function regression:
  the loss and empirical risk is just as for 'scalar-on-scalar regression'
  - We just have to implement new base-learners and
  - allow for different data input (e.g. matrices for curve observations)
- Function-on-function regression:

Implementation in FDboost (I)

Goal: Make use of the modular implementation of `mboost`

- Scalar-on-function regression:
  the loss and empirical risk is just as for 'scalar-on-scalar regression'
  - We just have to implement new base-learners and
  - allow for different data input (e.g. matrices for curve observations)
- Function-on-function regression:
  - Implement base-learner, which also vary in the direction of t

Implementation in `FDboost` (I)

Goal: Make use of the modular implementation of `mboost`

- Scalar-on-function regression:
  the loss and empirical risk is just as for 'scalar-on-scalar regression'
  - We just have to implement new base-learners and
  - allow for different data input (e.g. matrices for curve observations)
- Function-on-function regression:
  - Implement base-learner, which also vary in the direction of t
  - The loss function is now an integral

Implementation in `FDboost` (I)

Goal: Make use of the modular implementation of `mboost`

- Scalar-on-function regression:
  the loss and empirical risk is just as for 'scalar-on-scalar regression'
  - We just have to implement new base-learners and
  - allow for different data input (e.g. matrices for curve observations)
- Function-on-function regression:
  - Implement base-learner, which also vary in the direction of t
  - The loss function is now an integral
    - $\rightarrow$ Numerical integration scheme to approximate expected loss

Implementation in `FDboost` (II)

- Main fitting function:

  `FDboost(formula, timeformula, data, ...)`

- `timeformula`

  = `NULL` for scalar-on-function regression,
  = $\sim$ `bbs(t)` for function-on-function regression

- Some of the base-learners for functional data:

$z\beta(t)$	`bolsc(z)`
$f(z,t)$	`bbsc(z)`
$z_1 z_2 \beta(t)$	`bols(z1) %Xc% bols(z2)`
$\int_{\mathcal{S}} x(s)\beta(s,t)ds$	`bsignal(x, s = s)`
$x(t)\beta(t)$	`bconcurrent(x, s = s, time = t)`
$\int_{l(t)}^{u(t)} x(s)\beta(s,t)ds$	`bhist(x, s = s, time = t, limits = ...)`

Implementation in FDboost (II)

- Main fitting function:

  `FDboost(formula, timeformula, data, ...)`

- `timeformula`

  = `NULL` for scalar-on-function regression,

  = $\sim$ `bbs(t)` for function-on-function regression

- Some of the base-learners for functional data:

$z\beta(t)$	`bolsc(z) %O% bbs(t)`
$f(z,t)$	`bbsc(z) %O% bbs(t)`
$z_1 z_2 \beta(t)$	`bols(z1) %Xc% bols(z2) %O% bbs(t)`
$\int_{\mathcal{S}} x(s)\beta(s,t)ds$	`bsignal(x, s = s) %O% bbs(t)`
$x(t)\beta(t)$	`bconcurrent(x, s = s, time = t)`
$\int_{l(t)}^{u(t)} x(s)\beta(s,t)ds$	`bhist(x, s = s, time = t, limits = ...)`

Example: FDboost call

```
FDboost(EMG ~ 1 +
            brandomc(id, df = 5) +
            bhist(EEG, df = 20),
         timeformula = ~ bbs(t, df = 4),
         control = boost_control(mstop = 5000,
                                 trace = TRUE),
         data = data)
```

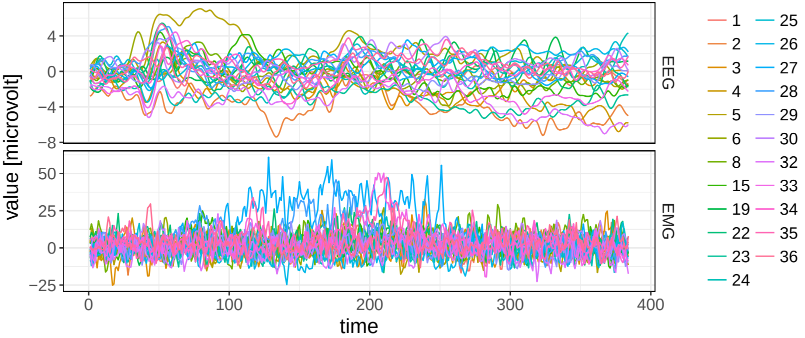
Case studies
Functional response

Emotion components data

Goal: Try to explain

- facial expressions (measured with EMG)
- by brain activity (measured with EEG)

$\rightarrow$ Function-on-function-regression



(Brockhaus et al., 2017)

Emotion components data

Model equation:

$$y_{\mathrm{EMG}}(t) = \beta_0(t) + x_{\mathrm{EEG}}(t)\beta_1(t) + \varepsilon(t)$$

- One-to-one relation between EEG and EMG $\rightarrow$ Concurrent effect

Emotion components data

Model equation:

$$y_{\mathrm{EMG}}(t) = \beta_0(t) + \int x_{\mathrm{EEG}}(s)\beta_1(s, t)\mathrm{d}s + \varepsilon(t)$$

- One-to-one relation between EEG and EMG $\rightarrow$ Concurrent effect
- Response time specific effect $\rightarrow$ Linear functional effect

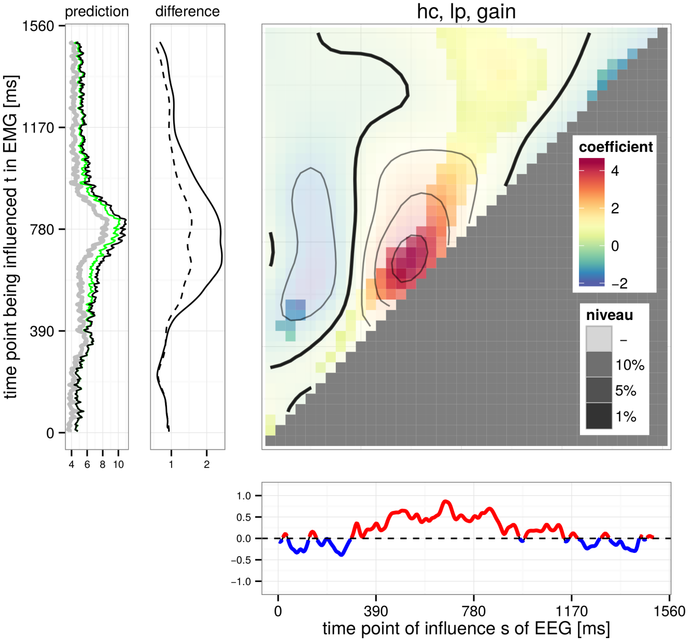
Emotion components data

Model equation:

$$y_{\mathrm{EMG}}(t) = \beta_0(t) + \int_0^{t-\delta} x_{\mathrm{EEG}}(s)\beta_1(s,t)\mathrm{d}s + \varepsilon(t)$$

- One-to-one relation between EEG and EMG $\rightarrow$ Concurrent effect
- Response time specific effect $\rightarrow$ Linear functional effect
- EMG can only be influenced by EEG activities in the past $\rightarrow$ Historical effect

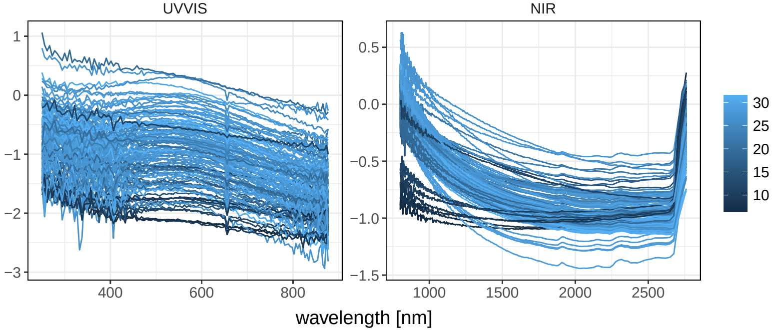
Results for more complex model



Case studies
Scalar response and functional covariates

Spectral data of fossil fuels

Goal: predict heat value y using the spectral measurements of NIR and UV spectra



(Brockhaus et al., 2015)

Data of fossil fuel: model

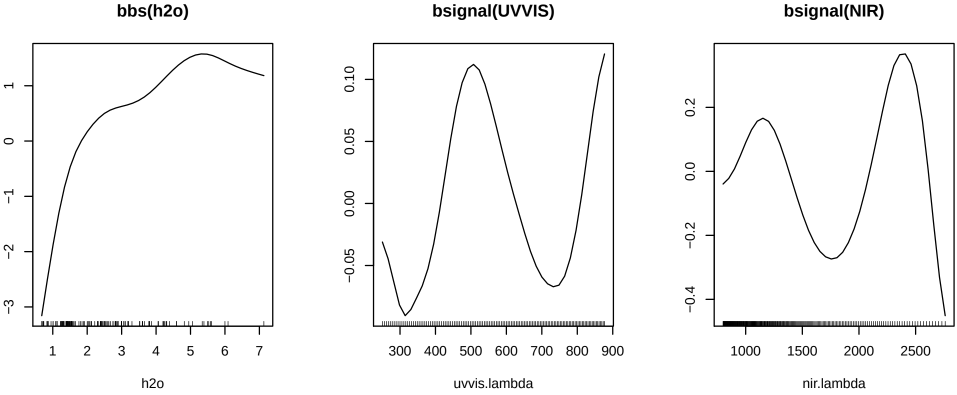
Model equation:

$$y = \beta_0 + f(z_{\mathrm{H2O}}) + \int_{\mathcal{S}_{\mathrm{NIR}}} x_{\mathrm{NIR}}(s_{\mathrm{NIR}})\beta_{\mathrm{NIR}}(s_{\mathrm{NIR}})\, ds_{\mathrm{NIR}} + \int_{\mathcal{S}_{\mathrm{UV}}} x_{\mathrm{UV}}(s_{\mathrm{UV}})\beta_{\mathrm{UV}}(s_{\mathrm{UV}})\, ds_{\mathrm{UV}} + \varepsilon,$$

- heat value y
- non-linear effect of water content (H2O)
- linear functional effect of NIR and UV spectrum

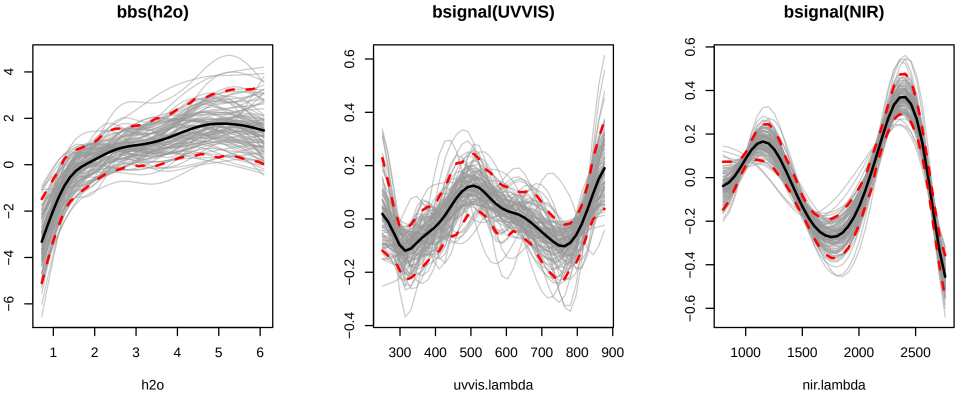
Data of fossil fuel: results

Estimated effects with stopping iteration chosen by 10 fold bootstrap



Data of fossil fuel: results

- estimated effects on 100 bootstrap samples (gray lines)
- point-wise median (black lines)
- point-wise 5 and 95% quantiles (dashed red lines)



Summary and discussion

Summary and discussion (I)

What is FDA?

- Measurement units are functions,
  i.e., curves, surfaces, trajectories,...
- Smooth data generating process
- Many observations of the same data generating process

- Mean, variance and covariance for functional data
- Functional counterparts for many methods from multivariate statistics

Summary and discussion (II)

Estimating functional regression models with `FDboost`

- LM, GLM, GAMLSS and quantile regression included
- variety of covariate effects,
  - (non-)linear effects of scalar covariates
  - linear effects of functional covariates, historical effects
  - interaction effects

Summary and discussion (II)

Estimating functional regression models with `FDboost`

- LM, GLM, GAMLSS and quantile regression included
- variety of covariate effects,
  - (non-)linear effects of scalar covariates
  - linear effects of functional covariates, historical effects
  - interaction effects
- estimation by component-wise gradient boosting
  - high dimensional data settings
  - data-driven variable selection
  - shrinkage of effects
- comprehensive implementation in R add-on package `FDboost`

References I

Vincent Bonhomme, Sandrine Picq, Cedric Gaucherel, and Julien Claude. Momocs: Outline analysis using R. *Journal of Statistical Software*, 56(13): 1–24, 2014. URL `http://www.jstatsoft.org/v56/i13/`.

Sarah Brockhaus and David Rügamer. *FDboost: Boosting Functional Regression Models*, 2017. R package version 0.3-0, Available at `http://CRAN.R-project.org/package=FDboost`.

Sarah Brockhaus, Fabian Scheipl, Torsten Hothorn, and Sonja Greven. The functional linear array model. *Statistical Modelling*, 15(3):279–300, 2015.

Sarah Brockhaus, Andreas Fuest, Andreas Mayr, and Sonja Greven. Signal regression models for location, scale and shape with an application to stock returns. arXiv preprint, arXiv:1605.04281, 2016a.

Sarah Brockhaus, Michael Melcher, Friedrich Leisch, and Sonja Greven. Boosting flexible functional regression models with a high number of functional historical effects. *Statistics and Computing*, 2016b. Accepted, DOI: `http://dx.doi.org/10.1007/s11222-016-9662-1`.

Sarah Brockhaus, David Rügamer, and Sonja Greven. Boosting Functional Regression Models with FDboost. *ArXiv e-print, arXiv:1705.10662*, May 2017.

References II

I. D. Currie, M. Durban, and P. H. C. Eilers. Generalized linear array models with applications to multidimensional smoothing. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(2):259–280, 2006.

Karen Fuchs, Fabian Scheipl, and Sonja Greven. Penalized scalar-on-functions regression with interaction term. *Computational Statistics & Data Analysis*, 81:38–51, 2015.

Kornelia Gentsch, Didier Grandjean, and Klaus R. Scherer. Coherence explored between emotion components: Evidence from event-related potentials and facial electromyography. *Biological Psychology*, 98(0):70 – 81, 2014.

Sonja Greven and Fabian Scheipl. A general framework for functional regression modelling. *Statistical Modelling*, 17(1-2):1–35, 2017.

Benjamin Hofner, Andreas Mayr, Nora Fenske, and Matthias Schmid. *gamboostLSS: Boosting Methods for GAMLSS Models*, 2017. R package version 2.0-0, Available at `http://CRAN.R-project.org/package=gamboostLSS`.

Torsten Hothorn, Peter Bühlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. *mboost: Model-Based Boosting*, 2017. R package version 2.8-1, Available at `http://CRAN.R-project.org/package=mboost`.

References III

Nicolai Meinshausen and Peter Bühlmann. Stability selection (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

Jeffrey S. Morris. Functional regression. *Annual Review of Statistics and Its Application*, 2(1):321–359, 2015.

James O Ramsay and Bernard W Silverman. *Functional Data Analysis*. Springer, New York, 2005.

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape (with discussion). *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554, 2005.

David Rügamer, Sarah Brockhaus, Kornelia Gentsch, Klaus Scherer, and Sonja Greven. Detecting synchronisation in EEG- and EMG-signals via boosted functional historical models. vArXiv e-print, arXiv:1609.06070, 2016.

Fabian Scheipl, Ana-Maria Staicu, and Sonja Greven. Functional additive mixed models. *Journal of Computational and Graphical Statistics*, 24(2): 477–501, 2015.

Fabian Scheipl, Jan Gertheiss, and Sonja Greven. Generalized functional additive mixed models. *Electronic Journal of Statistics*, 10(1):1455–1492, 2016.

References IV

Rajen D. Shah and Richard J. Samworth. Variable selection with error control: another look at stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(1):55–80, 2013.