Review of Functional Partial Least Squares

Application to Spectral Misalignment

Slides by Rory Samuels

Functional Linear Regression Model

Suppose we have sample of scalar valued response variables $y_i \in \mathbb{R}$ and functional valued predictors $x_i(t) \in L^2([a,b])$, for $i = 1, .., n$. The functional linear regression model (FLM) is given by:

$$y_i = \beta_0 + \int_a^b x_i(t)\beta(t)dt + \epsilon_i,$$

where β_0 is the intercept, $\beta(t)$ is the functional coefficient, and $\epsilon_i \sim N(0, \sigma^2)$.

- Note: for notational simplicity we assume $\beta_0 = 0$.

Functional Partial Least Squares (FPLS)

Given weight functions $w_1(t), ..., w_{k-1}(t)$, the kth weight function is obtained via

$$\arg\max_{w(t)} \mathsf{Cov}^2 \left(y, \int_a^b x(t)w(t)dt \right).$$

subject to:

$$\mathsf{Cov} \left(\int_a^b x(t)w_j(t)dt, \int_a^b x(t)w(t)dt \right) = 0 \quad \text{for} \quad j = 1, ..., k-1,$$

$$\text{and } ||w(t)||_2^2 = 1.$$

Basis Approximation for Weight Functions

Let $\mathbf{B}(t) = (B_1(t), ..., B_{M+d}(t))'$ be a vector of $M + d$ B-spline basis functions of degree d. We can approximate the jth weight function by

$$w_j(t) \approx \mathbf{b}_j' \mathbf{B}(t),$$

where $\mathbf{b}_j$ is a vector of $M + d$ basis coefficients.

- If we let $u_r = \int_a^b x(t) B_r(t) dt$ and $\mathbf{u} = (u_1, ..., u_{M+d})'$ then

$$\int_a^b x(t) w_j(t) dt \approx \mathbf{u}' \mathbf{b}_j$$

Basis Approximation for FPLS

Given $\mathbf{b}_1, ..., \mathbf{b}_{k-1}$, the kth weight vector is obtained via

$$\arg\max_{\mathbf{b}} \mathsf{Cov}^2\left(y, \mathbf{u}'\mathbf{b}\right).$$

subject to:

$$\mathsf{Cov}\left(\mathbf{u}'\mathbf{b}_j, \mathbf{u}'\mathbf{b}\right) = 0 \quad \text{for} \quad j = 1, ..., k-1,$$

$$\text{and } ||\mathbf{b}'\mathbf{V}\mathbf{b}||_2^2 = 1.^{1}$$

[1] $\mathbf{V}$ is the pos. def. matrix of inner products between all pairs of basis functions.

FPLS (Empirically)

Let $\mathbf{y}$ be a vector of n observations of the response, and $\mathbf{U}$ be the $n \times (M + d)$ matrix with elements

$$\mathbf{U}_{(ij)} = \int_a^b x_i(t) B_j(t) dt.$$

- Finding optimal $\mathbf{b}_j$'s is equivalent to performing classical PLS with response $\mathbf{y}$ and covariates $\mathbf{U}$.
- Can be done efficiently with existing algorithms (e.g. SIMPLS or NIPALS).

FPLS Coefficient

Let $\mathbf{R}$ be the $(M + d) \times K$ matrix whose columns are the first K empirical weight vectors $\hat{\mathbf{b}}_1, ..., \hat{\mathbf{b}}_K$.

- U Scores: $\mathbf{T} = \mathbf{UR}$
- Y loadings: $\mathbf{q} = (\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{y}$

The estimated functional coefficient is then

$$\hat{\beta}_{FPLS}(t) = (\mathbf{Rq})'\mathbf{B}(t)$$

Starting from Discrete Observations

The key to functional partial least squares is obtaining

$$\mathbf{U}_{(ij)} = \int_a^b x_i(t) B_j(t) dt, \quad i = 1, ..., n, \quad j = 1, ..., M + d.$$

- In practice, we observe p discrete points along each $x_i(t)$ (possibly with noise).

Numerical Approximation

- If we have a dense observation grid, and negligible instrument noise, we can approximate $\mathbf{U}_{(ij)}$ by

$$\mathbf{U}_{(ij)} \approx \frac{b-a}{p} \sum_{k=1}^{p} x_i(t_k) B_j(t_k).$$

Basis Expansion for Data

- Alternatively, we can expand each observation onto a set of suitable basis functions:

$$x_i(t) \approx \mathbf{c}_i' \mathbf{B}^x(t),$$

where $\mathbf{B}^x(t)$ is a vector of $M_x + d$ B-spline basis functions and $\mathbf{c}_i$ is a vector of $M_x + d$ basis coefficients. In this case

$$\mathbf{U}_{(ij)} \approx \mathbf{C}\boldsymbol{\Theta},$$

where $\mathbf{C}$ is an $n \times (M_x + d)$ matrix of basis coefficients and $\boldsymbol{\Theta}$ is an $(M_x + d) \times (M + d)$ matrix with elements

$$\boldsymbol{\Theta}_{(ij)} = \int_a^b B_i^x(t) B_j(t) dt.$$

Example I: Generated Responses

We generated $n = 500$ scalar responses from

$$y_i = \int_0^1 x_i(t)\beta(t)dt + \epsilon_i$$

- $x_i(t)$: random linear combinations of cubic B-spline basis functions[2]
- $\beta(t) = 10(t-1)^2 + 30\cos(4\pi t^3)$
- $\epsilon_i \sim N(0, \sigma_\epsilon^2)$[3]

[2] The basis functions were defined over 50 knots and all coefficients were generated from a standard normal distribution.

[3] The error variance σ_ϵ^2 was chosen such that the signal-to-noise ratio was 5.

Example I: Generated Predictors

To simulate misalignment, we sampled each $x_i(t)$ along two observation grids, G_A and G_B, of length 425 and 150 respectively.

- G_A: $t = 0, .0024, .0048, ..., 1$
- G_B: $t = 0, .0068, .0136, ..., 1$

The final data-set consisted of y_i and corresponding discrete observations of $x_i(t)$ on both G_A and G_B, for $i = 1, ..., 500$.

Example I: Generated Predictors

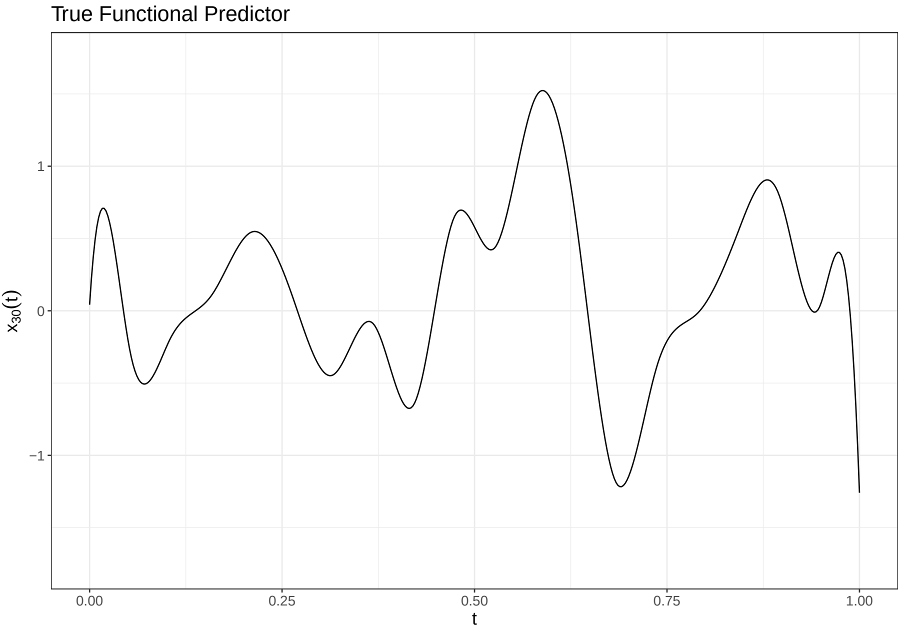
To simulate misalignment, we sampled each $x_i(t)$ along two observation grids, G_A and G_B, of length 425 and 150 respectively.

- G_A: $t = 0, .0024, .0048, ..., 1$
- G_B: $t = 0, .0068, .0136, ..., 1$

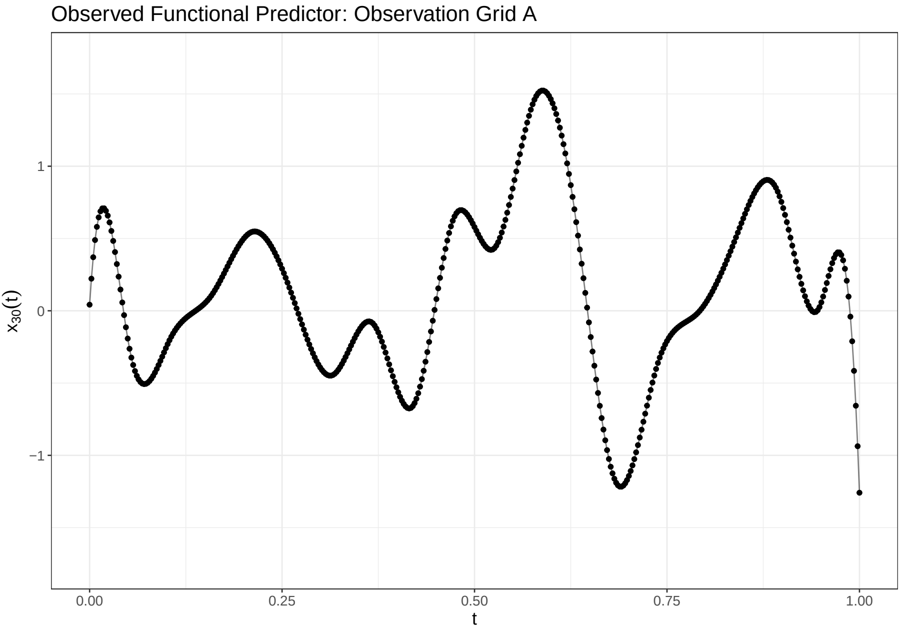
The final data-set consisted of y_i and corresponding discrete observations of $x_i(t)$ on both G_A and G_B, for $i = 1, ..., 500$.

- Goal: predict y from $x(t)$ observed on G_B, using a model trained with $x(t)$ observed on G_A.
  - $80/20$ train/test split.

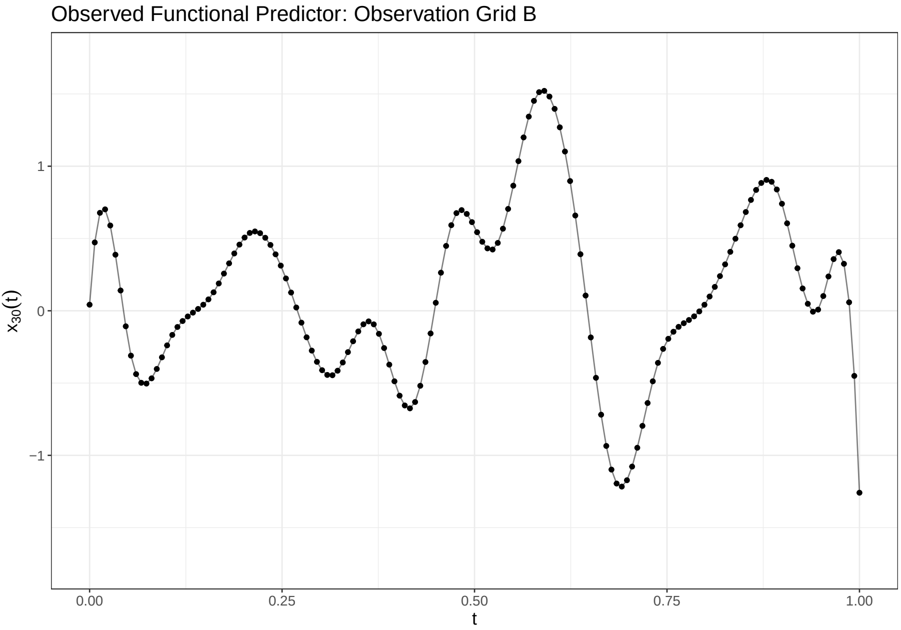
Example I: Misaligned Grids



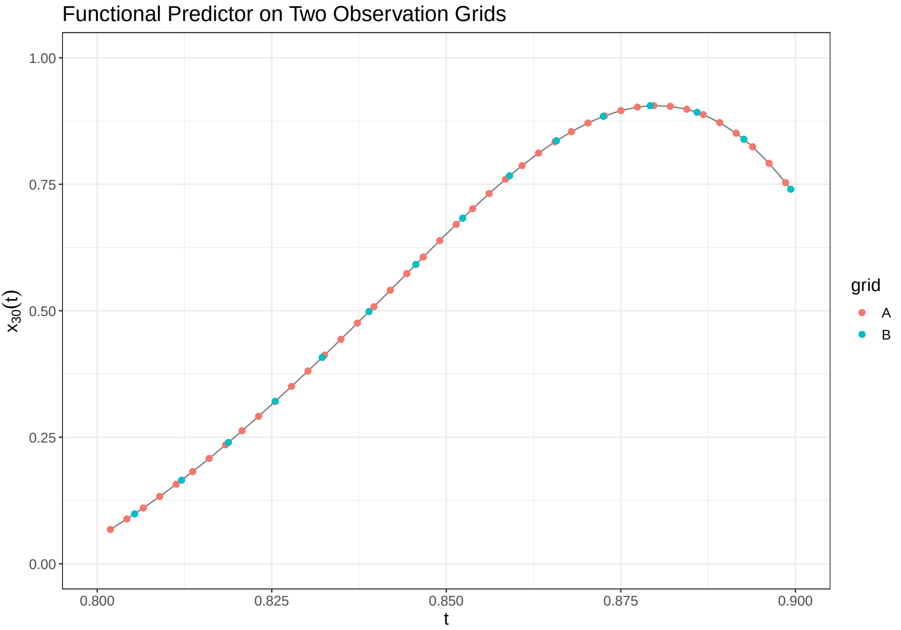
Example I: Misaligned Grids



Example I: Misaligned Grids



Example I: Misaligned Grids



Example I: Two Approaches

- Goal: predict y from $x(t)$ observed on G_B, using a model trained with $x(t)$ observed on G_A.
  - $80/20$ train/test split.

Classical PLS Approach:

- Obtain PLS coefficients $\hat{\boldsymbol{\beta}}_A$ using y^{train} and $x^{train}(t)$ on G_A
- Select PLS coefficients closest to points on G_B, $\hat{\boldsymbol{\beta}}_B$
- Predict y^{test} using observations of $x^{test}(t)$ on G_B and $\hat{\boldsymbol{\beta}}_B$

Example I: Two Approaches

- Goal: predict y from $x(t)$ observed on G_B, using a model trained with $x(t)$ observed on G_A.
  - $80/20$ train/test split.

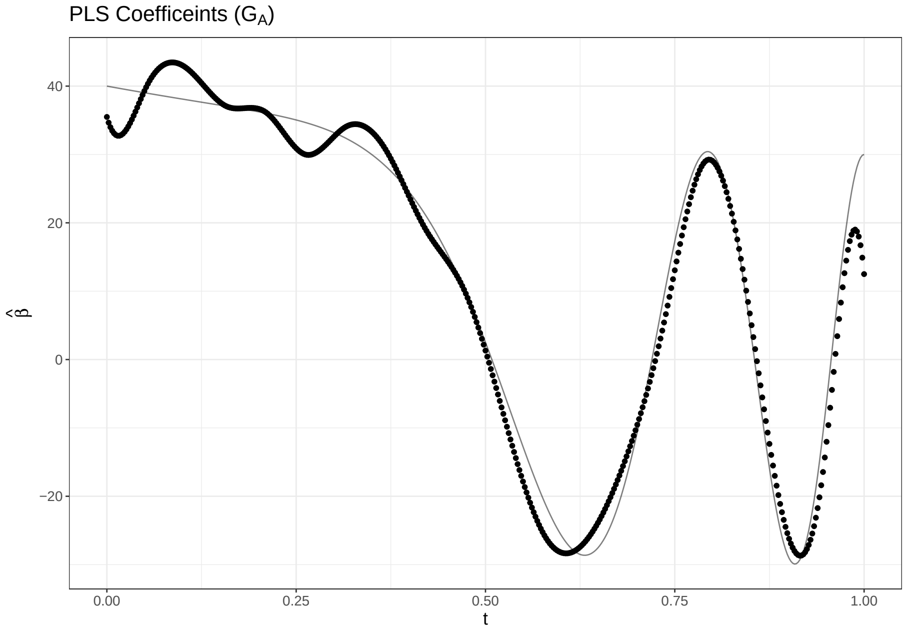
Classical PLS Approach:

- Obtain PLS coefficients $\hat{\boldsymbol{\beta}}_A$ using y^{train} and $x^{train}(t)$ on G_A
- Select PLS coefficients closest to points on G_B, $\hat{\boldsymbol{\beta}}_B$
- Predict y^{test} using observations of $x^{test}(t)$ on G_B and $\hat{\boldsymbol{\beta}}_B$

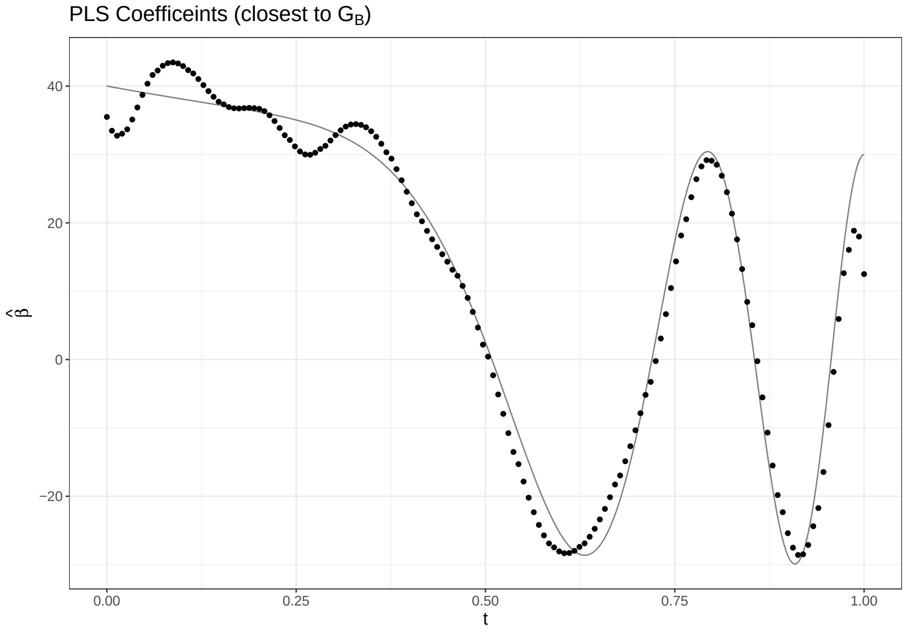
Functional PLS approach:

- Obtain $\hat{\beta}_{FPLS}(t)$ using observations of $x^{train}(t)$ on G_A
- Predict y^{test} using observations of $x^{test}(t)$ on G_B and $\hat{\beta}(t)$.

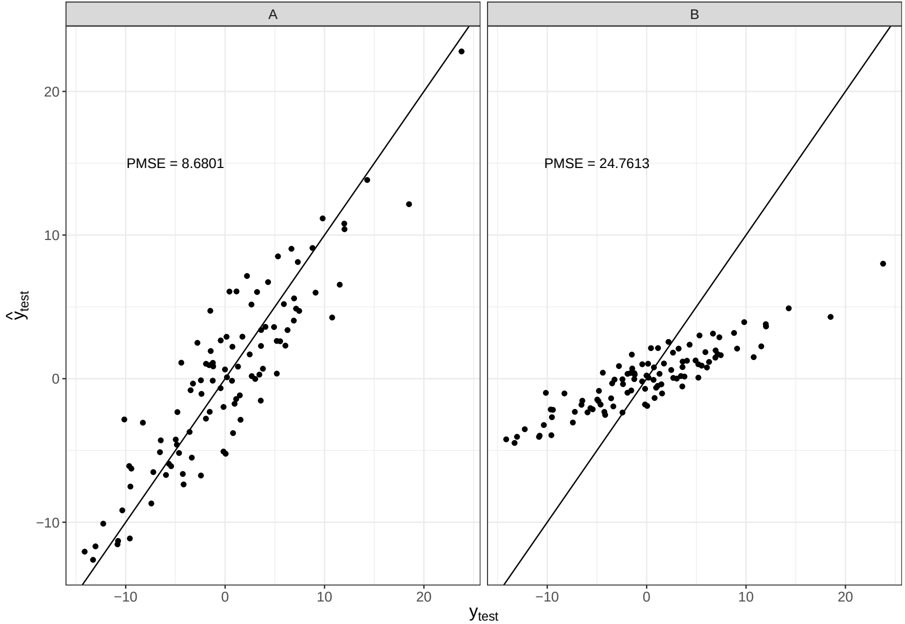
Example I: Classical PLS



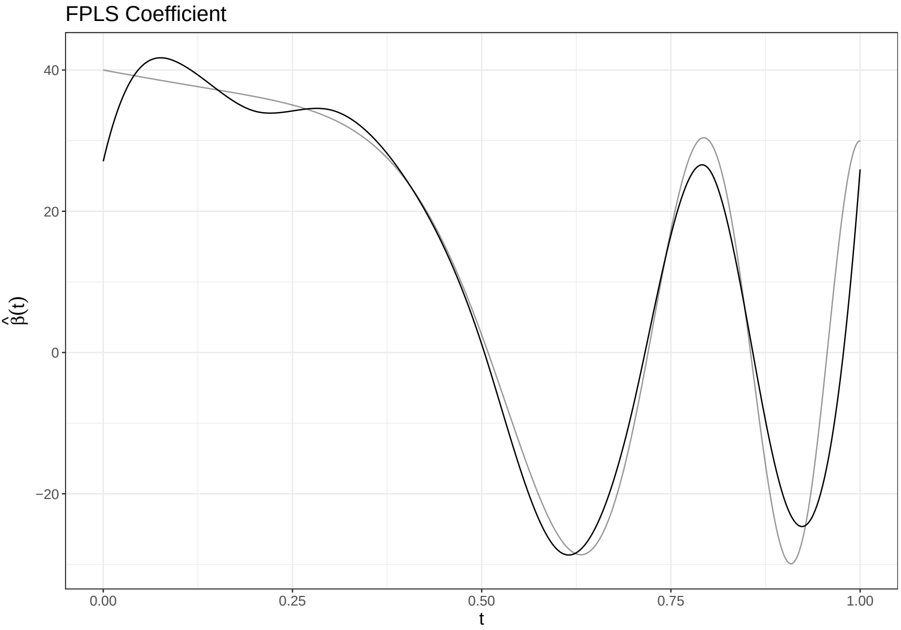
Example I: Classical PLS



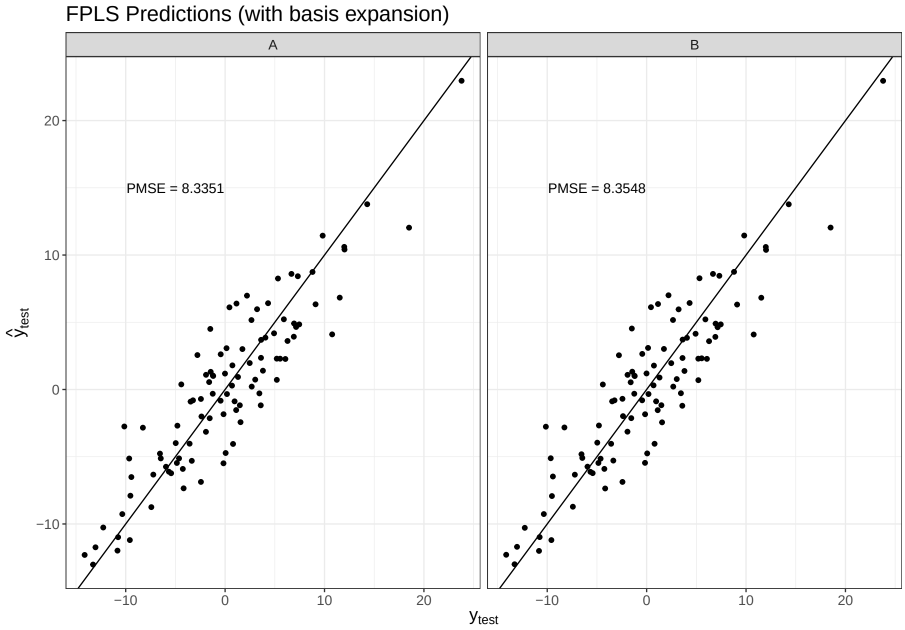
Example I: Classical PLS



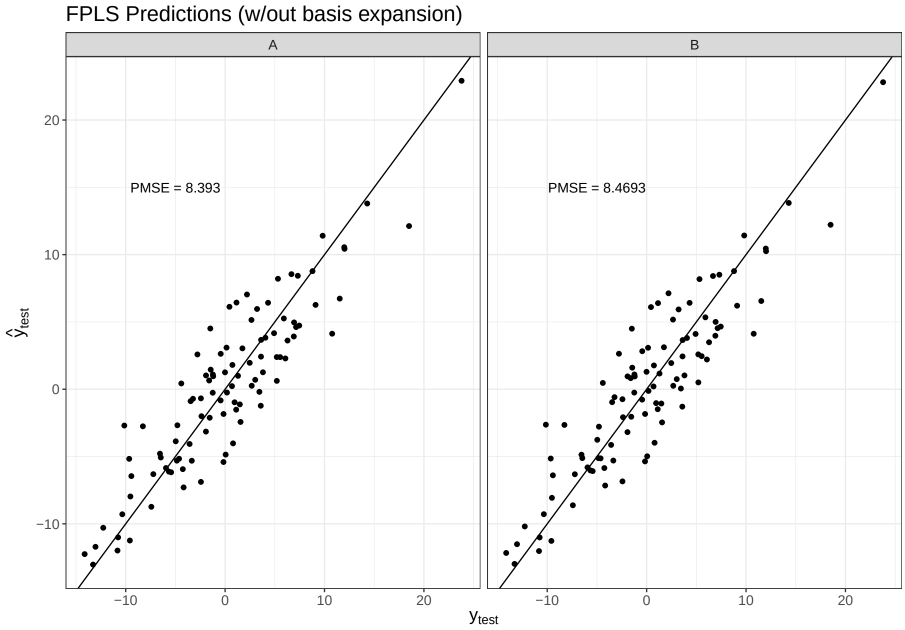
Example I: Functional PLS



Example I: Functional PLS



Example I: Functional PLS



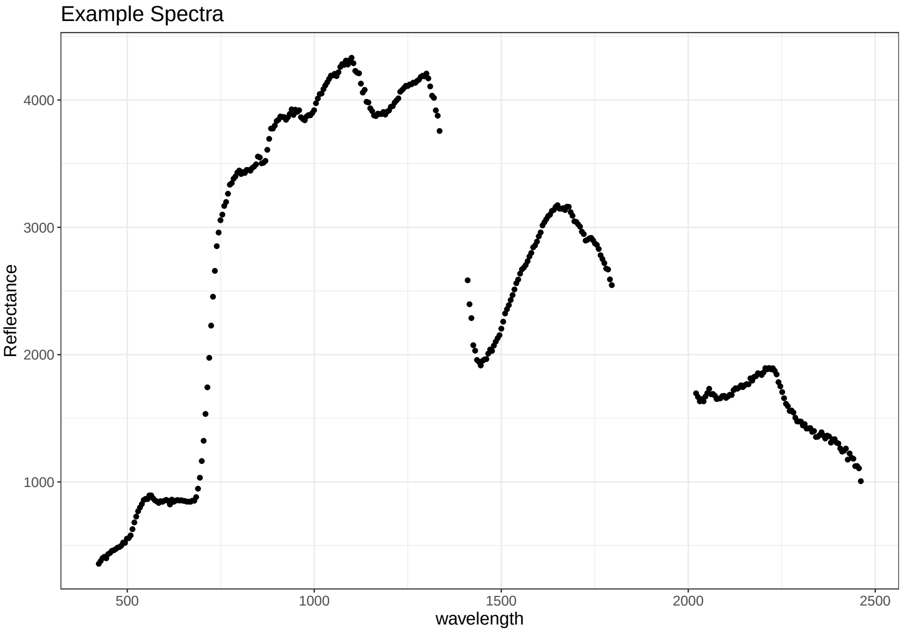
Example II: AOP Crown Data

We applied the same method to the AOP Crown data to predict `d15N` from spectra. After joining the site trait data and spectra by `SampleSiteID`, and removing both "bad bands" and NA observations we had:

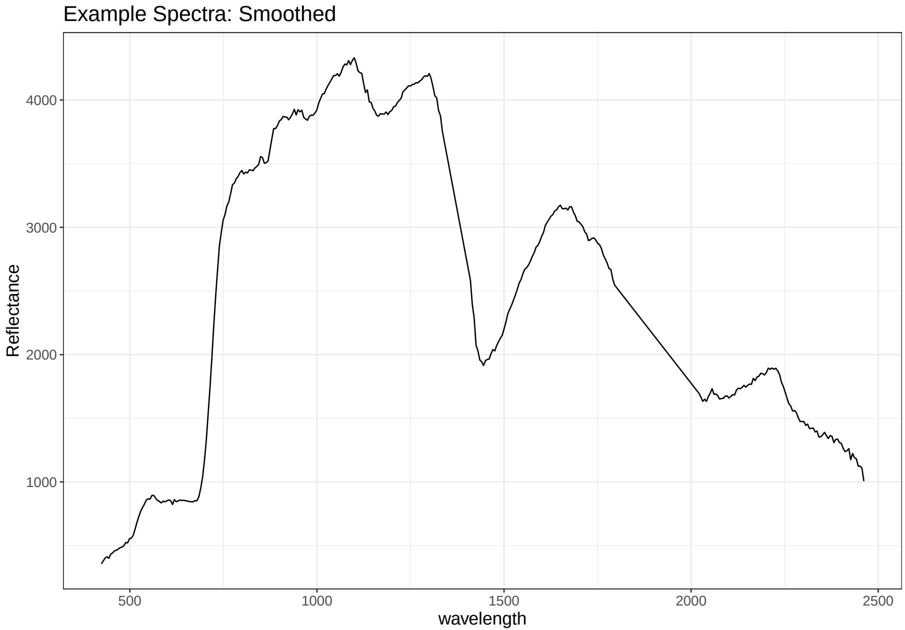
- $n = 2515$ observations
- $p_A = 350$ spectral points per spectra.

To simulate spectral misalignment, we expanded the spectra onto a set of 52 cubic B-splines and sampled along an observation grid of $p_B = 200$ points.

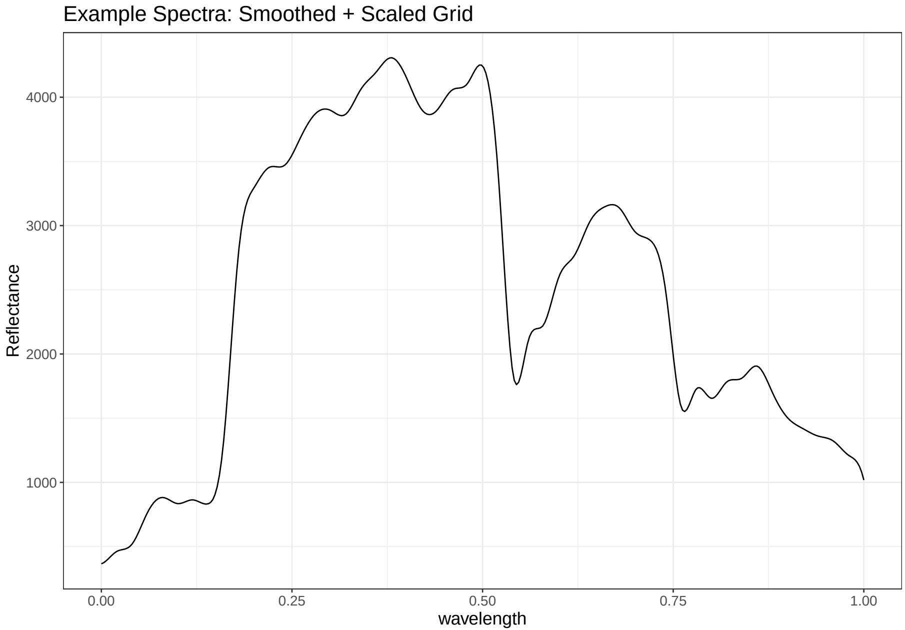
Example II: Spectra



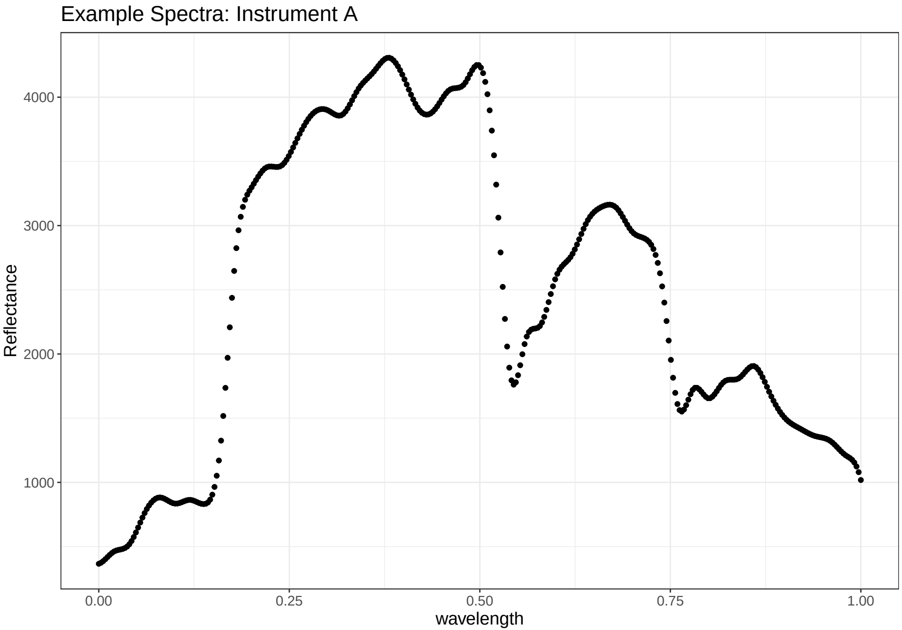
Example II: Spectra



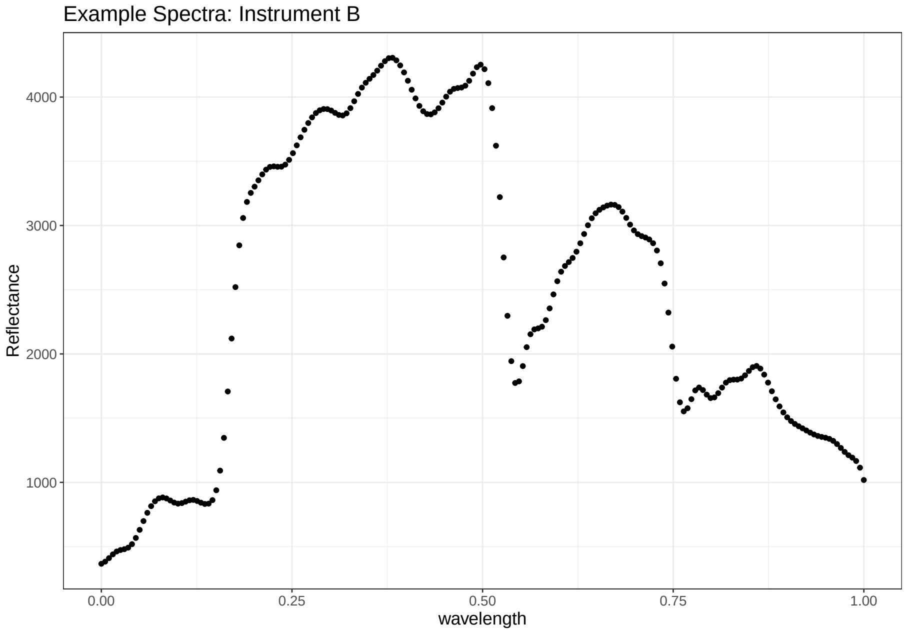
Example II: Spectra



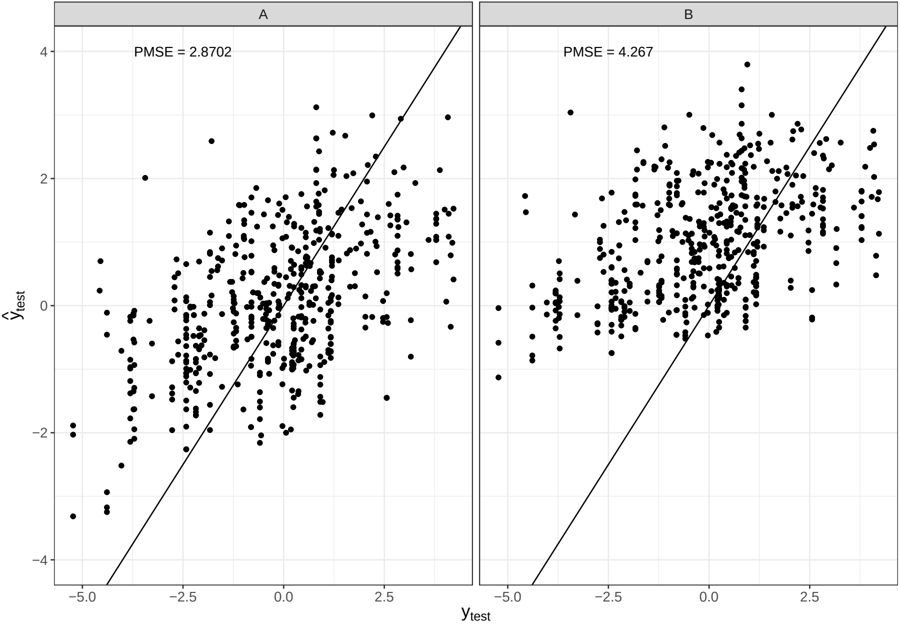
Example II: Spectra



Example II: Spectra



Classical PLS Approach



Functional PLS Approach