

Package: FSHybridPLS (via r-universe)

September 23, 2026

Title Hybrid Penalized Partial Least Squares for Mixed Data

Version 0.1.0

Description Fits Penalized Partial Least Squares (PLS) regression when predictors are hybrid objects that combine functional curves (infinite-dimensional 'fda' objects) and scalar covariates (finite-dimensional numeric matrices). The package treats a hybrid predictor as an element of a product Hilbert space formed by the functional and Euclidean components, and implements the arithmetic (addition, scalar multiplication, and inner products, including roughness-penalized inner products) needed to run penalized PLS directly in that space. The algorithm extracts latent components that maximize covariance with a scalar response while penalizing roughness of the estimated functional coefficient curves. Helpers are included for constructing hybrid predictors, two-step within- and between-modality normalization, train/test splitting, synthetic data generation, cross-validated component selection, and prediction. The method is described in Mun and Jang (2026) <doi:10.48550/arXiv.2601.16364>.

License MIT + file LICENSE

URL <https://github.com/Jong-Min-Moon/FSHybridPLS>

BugReports <https://github.com/Jong-Min-Moon/FSHybridPLS/issues>

Imports fda (>= 6.1.3), stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/checktor/allow print_cat_usage

Config/checktor/acronyms PLS

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

NeedsCompilation no

VignetteBuilder knitr
Author Jongmin Mun [aut, cre, cph]
Maintainer Jongmin Mun <jongmin.mun@marshall.usc.edu>
Config/pak/sysreqs make
Repository https://jongmin-moon.r-universe.dev
Date/Publication 2026-09-14 20:58:33 UTC
RemoteUrl https://github.com/jongmin-moon/fshybridpls
RemoteRef HEAD
RemoteSha 97563d00c5afc97c8bc96bcb3bd596db3a126c52

Contents

create_idx_kfold	2
cv_fit_hybridPLS	3
fit.hybridPLS	4
inprod.predictor_hybrid	5
inprod_pen.predictor_hybrid	6
load_and_preprocess_kidney_data	6
predict.hybridPLS	7
predictor_hybrid	8
simulate_hybrid_data	9
split_all	10
split_and_normalize.all	11
Index	13

create_idx_kfold	<i>Generate K-Fold Cross-Validation Indices</i>
------------------	---

Description

Generate K-Fold Cross-Validation Indices

Usage

create_idx_kfold(n_sample, n_fold)

Arguments

- n_sample Integer. Total number of samples.
- n_fold Integer. Number of folds (>= 2).

Value

A list of length n_fold; each element has idx_train and idx_valid.

Examples

```
set.seed(1)
folds <- create_idx_kfold(20, 4)
length(folds)
length(folds[[1]]$idx_valid)
```

cv_fit_hybridPLS

*Cross-Validated Hybrid Penalized PLS***Description**

Fits `fit_hybridPLS()` on K training folds and records mean validation RMSE for each number of components. Useful for choosing `n_iter`.

Usage

```
cv_fit_hybridPLS(W, y, n_iter, lambda, n_fold = 5, seed = NULL)
```

Arguments

<code>W</code>	A predictor_hybrid object.
<code>y</code>	Numeric response vector.
<code>n_iter</code>	Maximum number of components to consider.
<code>lambda</code>	Numeric smoothing-parameter vector (length <code>W\$n_functional</code>).
<code>n_fold</code>	Number of CV folds (default 5).
<code>seed</code>	Optional seed for fold assignment.

Value

A list with:

rmse_by_component Mean validation RMSE for components 1:`n_iter`.

best_n_iter Component count with smallest mean RMSE.

fold_rmse Matrix of fold-wise RMSE (`n_fold` x `n_iter`).

Examples

```
set.seed(1)
sim <- simulate_hybrid_data(n = 40, n_basis = 5)
cv <- cv_fit_hybridPLS(sim$W, sim$y, n_iter = 3, lambda = 1e-3, n_fold = 3)
cv$best_n_iter
```

fit.hybridPLS

*Fit Hybrid Penalized PLS***Description**

Runs the iterative Hybrid Penalized Partial Least Squares algorithm. If `validation_data` is provided, RMSE on the validation set is stored for each extracted component.

Usage

```
fit.hybridPLS(W, y, n_iter, lambda, validation_data = NULL)
```

```
fit_hybridPLS(W, y, n_iter, lambda, validation_data = NULL)
```

```
## S3 method for class 'hybridPLS'
print(x, ...)
```

Arguments

<code>W</code>	A predictor_hybrid object (training predictors).
<code>y</code>	Numeric vector of training responses (length <code>W\$n_sample</code>).
<code>n_iter</code>	Integer number of PLS components to extract (≥ 1).
<code>lambda</code>	Numeric vector of smoothing parameters, one per functional predictor (length(<code>lambda</code>) == <code>W\$n_functional</code>).
<code>validation_data</code>	Optional list with elements <code>W_test</code> (predictor_hybrid) and <code>y_test</code> (numeric vector).
<code>x</code>	A hybridPLS object.
<code>...</code>	Unused.

Value

An object of class `hybridPLS`, a list with:

rho List of score vectors (length `n_iter`).

xi List of weight directions as single-sample predictor_hybrid objects.

beta List of cumulative regression directions; `beta[[L]]` uses the first `L` components.

W List of deflated predictors; `W[[1]]` is the input, `W[[1+1]]` is after extracting component 1.

delta List of predictor loadings.

nu List of response loadings (scalars).

n_iter, lambda Fitting settings.

validation_rmse Numeric vector of validation RMSE by component, or `NULL`.

Examples

```

set.seed(1)
sim <- simulate_hybrid_data(n = 40, n_functional = 1, n_scalar = 2, n_basis = 5)
fit <- fit_hybridPLS(sim$W, sim$y, n_iter = 2, lambda = 1e-3)
fit
pred <- predict(fit, sim$W, n_components = 2)
cor(pred, sim$y)

```

inprod.predictor_hybrid

Algebraic Inner Product for Hybrid Objects

Description

Uses pre-computed Gram matrices to calculate hybrid inner products without numerical integration.

Usage

```
inprod.predictor_hybrid(xi_1, xi_2 = NULL)
```

```
inprod_predictor_hybrid(xi_1, xi_2 = NULL)
```

Arguments

xi_1	A predictor_hybrid object.
xi_2	Another predictor_hybrid object (optional). If NULL, the inner product of xi_1 with itself is computed.

Value

A numeric scalar or vector of inner products. When one argument has a single sample and the other has N samples, returns a length-N vector (broadcasted projection).

Examples

```

set.seed(1)
sim <- simulate_hybrid_data(n = 20, n_basis = 5)
fit <- fit_hybridPLS(sim$W, sim$y, n_iter = 1, lambda = 1e-3)
scores <- inprod_predictor_hybrid(sim$W, fit$xi[[1]])
length(scores)

```

inprod_pen.predictor_hybrid

Penalized Inner Product for Hybrid Objects

Description

Computes the unpenalized hybrid inner product plus roughness penalties on functional components:
 $\langle \xi_1, \xi_2 \rangle + \sum_k \lambda_k \langle D^m \xi_{1k}, D^m \xi_{2k} \rangle$.

Usage

```
inprod_pen.predictor_hybrid(xi_1, xi_2 = NULL, lambda)
```

```
inprod_pen_predictor_hybrid(xi_1, xi_2 = NULL, lambda)
```

Arguments

xi_1	A predictor_hybrid object.
xi_2	Another predictor_hybrid object (optional; defaults to xi_1).
lambda	Non-negative numeric vector of length xi_1\$n_functional.

Value

A numeric scalar (or vector under broadcasting), the penalized inner product.

Examples

```
set.seed(1)
sim <- simulate_hybrid_data(n = 20, n_basis = 5)
fit <- fit_hybridPLS(sim$W, sim$y, n_iter = 2, lambda = 1e-3)
inprod_pen_predictor_hybrid(fit$xi[[1]], fit$xi[[1]], lambda = 1e-3)
```

load_and_preprocess_kidney_data

Load and Preprocess Kidney Data for FSHybridPLS

Description

Processes a raw kidney study data frame into a predictor_hybrid object and a scaled response. The input is expected to contain at least columns ID, Study, and renogram_value, plus clinical covariates used for the response and scalar predictors (accessed by the positional columns described in the paper workflow).

Usage

```
load_and_preprocess_kidney_data(kidney_df, n_basis = 20)
```

Arguments

kidney_df Data frame with kidney study records.

n_basis Integer. Number of B-spline basis functions (default 20).

Details

This helper is application-specific and requires user-supplied data; it is not needed for the core Hybrid PLS algorithm.

Value

A list with:

W A predictor_hybrid object.

y Min-max scaled numeric response in $[0, 1]$.

Examples

```
try(load_and_preprocess_kidney_data(data.frame(x = 1)), silent = TRUE)
```

predict.hybridPLS	<i>Predict from a Hybrid PLS Model</i>
-------------------	--

Description

Computes fitted or predicted responses using the cumulative coefficient direction $\beta[[n_components]]$.

Usage

```
## S3 method for class 'hybridPLS'
predict(object, newdata = NULL, n_components = object$n_iter, ...)
```

Arguments

object A hybridPLS object from [fit_hybridPLS\(\)](#).

newdata A predictor_hybrid object. If omitted, predictions use the training predictors stored as `object$W[[1]]`.

n_components Number of components to use (default: all).

... Unused.

Value

Numeric vector of predictions (length newdata\$n_sample).

Examples

```
set.seed(1)
sim <- simulate_hybrid_data(n = 30, n_basis = 5)
fit <- fit_hybridPLS(sim$W, sim$y, n_iter = 2, lambda = 1e-3)
head(predict(fit, sim$W))
```

predictor_hybrid	<i>Create a predictor_hybrid object</i>
------------------	---

Description

Constructs an S3 object of class predictor_hybrid that stores scalar covariates and functional predictors in a joint representation, along with precomputed Gram and roughness-penalty matrices used by Hybrid PLS.

Usage

```
predictor_hybrid(Z, functional_list, eval_point, penalty_order = 2)
```

Arguments

Z	Numeric matrix (n_sample x n_scalar).
functional_list	List of 'fd' objects.
eval_point	Evaluation points.
penalty_order	Integer. Derivative order for roughness penalty (default 2).

Value

An object of class predictor_hybrid, a named list with:

Z	Scalar predictor matrix (n_sample x n_scalar).
functional_list	List of fd objects for the functional predictors.
gram_list	List of Gram (inner-product) matrices for each functional basis.
gram_deriv_list	List of roughness-penalty matrices for each functional basis.
eval_point	Numeric vector of evaluation points for the functional domain.
n_basis_list	Number of basis functions for each functional predictor.
n_sample	Number of samples.
n_functional	Number of functional predictors.
n_scalar	Number of scalar predictors.

Examples

```

library(fda)
# Create basis and functional data
fbasis <- create.bspline.basis(c(0, 1), 5)
coefs <- matrix(rnorm(15), 5, 3)
fd_obj <- fd(coefs, fbasis)

# Scalar data
Z_mat <- matrix(rnorm(9), 3, 3)

# Construct hybrid predictor
W <- predictor_hybrid(
  Z = Z_mat,
  functional_list = list(fd_obj),
  eval_point = seq(0, 1, 0.1)
)
class(W)
W$n_sample
W$n_functional
W$n_scalar

```

simulate_hybrid_data *Simulate Hybrid Functional-Scalar Regression Data*

Description

Generates synthetic hybrid predictors (functional curves + scalar covariates) and a scalar response useful for examples and testing.

Usage

```

simulate_hybrid_data(
  n = 50,
  n_functional = 1,
  n_scalar = 3,
  n_basis = 7,
  n_eval = 51,
  signal_to_noise = 3,
  seed = NULL
)

```

Arguments

n	Number of samples.
n_functional	Number of functional predictors.
n_scalar	Number of scalar predictors.
n_basis	Number of B-spline basis functions per functional predictor.

n_eval Number of evaluation grid points on $[0, 1]$.
signal_to_noise Approximate signal-to-noise ratio for the response.
seed Optional RNG seed.

Value

A list with:

W A predictor_hybrid object.

y Numeric response vector of length n .

beta_true Single-sample predictor_hybrid used to generate the linear signal.

Examples

```

set.seed(1)
sim <- simulate_hybrid_data(n = 25, n_functional = 1, n_scalar = 2, n_basis = 5)
class(sim$W)
length(sim$y)

```

split_all

Split Hybrid Predictor and Response Data

Description

Randomly partitions the hybrid predictor object and response vector into training and testing sets based on a given ratio.

Usage

```
split_all(W_hybrid, response, train_ratio)
```

Arguments

W_hybrid A predictor_hybrid object.
response A numeric (or factor) vector of length $W_hybrid\$n_sample$.
train_ratio Proportion of samples for training (strictly between 0 and 1).

Value

A named list with:

predictor_train, predictor_test predictor_hybrid objects.

response_train, response_test Response vectors for each split.

train_idx, test_idx Integer indices used for the split.

Examples

```
set.seed(1)
sim <- simulate_hybrid_data(n = 30, n_basis = 5)
parts <- split_all(sim$W, sim$y, train_ratio = 0.7)
length(parts$response_train) > 0
length(parts$response_test) > 0
inherits(parts$predictor_train, "predictor_hybrid")
```

split_and_normalize.all

Split and Preprocess Hybrid Data

Description

Full preprocessing pipeline: train/test split, within-modality standardization for functional and scalar predictors, between-modality variance balancing, and response standardization (train statistics applied to test).

Usage

```
split_and_normalize.all(W_hybrid, response, train_ratio)

split_and_normalize_all(W_hybrid, response, train_ratio)
```

Arguments

<code>W_hybrid</code>	A <code>predictor_hybrid</code> object.
<code>response</code>	Numeric response vector (length <code>W_hybrid\$n_sample</code>).
<code>train_ratio</code>	Proportion of samples used for training (strictly between 0 and 1).

Value

A list with:

predictor_train, predictor_test Normalized `predictor_hybrid` objects.

response_train, response_test Standardized responses.

details Intermediate normalization objects and split indices.

Examples

```
set.seed(1)
sim <- simulate_hybrid_data(n = 40, n_basis = 5)
prep <- split_and_normalize_all(sim$W, sim$y, train_ratio = 0.7)
fit <- fit_hybridPLS(
  prep$predictor_train,
  prep$response_train,
```

```
n_iter = 2,  
lambda = 1e-3,  
validation_data = list(  
  W_test = prep$predictor_test,  
  y_test = prep$response_test  
)  
)  
fit$validation_rmse
```

Index

`create_idx_kfold`, [2](#)
`cv_fit_hybridPLS`, [3](#)

`fit_hybridPLS`, [4](#)
`fit_hybridPLS (fit_hybridPLS)`, [4](#)
`fit_hybridPLS()`, [3](#), [7](#)

`inprod.predictor_hybrid`, [5](#)
`inprod_pen.predictor_hybrid`, [6](#)
`inprod_pen_predictor_hybrid`
 (`inprod_pen.predictor_hybrid`),
 [6](#)
`inprod_predictor_hybrid`
 (`inprod.predictor_hybrid`), [5](#)

`load_and_preprocess_kidney_data`, [6](#)

`predict_hybridPLS`, [7](#)
`predictor_hybrid`, [8](#)
`print_hybridPLS (fit_hybridPLS)`, [4](#)

`simulate_hybrid_data`, [9](#)
`split_all`, [10](#)
`split_and_normalize.all`, [11](#)
`split_and_normalize_all`
 (`split_and_normalize.all`), [11](#)