Package 'SmoothPLS'

May 7, 2026

Type Package

Title Partial Least-Squares Algorithm for Categorical and Scalar Functional Data

Version 0.1.5

Date 2026-04-10

Description Performs the Partial Least-Squares ('PLS') algorithm for functional data through the concept of active area integration. This approach builds upon the basis expansion methods for functional 'PLS' regression described in Aguilera et al. (2010) <doi:10.1016/j.chemolab.2010.09.007>. The package seamlessly handles both Scalar Functional Data ('SFD') and Categorical Functional Data ('CFD'), providing interpretable regression curves even for discrete state changes. It was developed during a PhD thesis between 'DECATHLON' and French research institute 'INRIA' 2022-2026. The 'SmoothPLS' method does not directly decompose the data into a basis; rather, it assumes the data is known as precisely as desired, and for every 'PLS' component, the weight functions are decomposed into the basis. For both single-state and multi-state 'CFD' as well as 'SFD', the algorithm is implemented for a scalar response. To provide a baseline, a naive 'PLS' method on time-value functions and standard Functional 'PLS' are also implemented.

License MIT + file LICENSE

URL https://github.com/FrancoisBassac/SmoothPLS, https://FrancoisBassac.github.io/SmoothPLS/

BugReports https://github.com/FrancoisBassac/SmoothPLS/issues

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Imports cfda (>= 0.12.1), dplyr (>= 1.1.4), fda (>= 6.2.0), ggplot2 (>= 3.5.1), MASS (>= 7.3-64), mgcv (>= 1.9-1), pls (>= 2.8-5), pracma (>= 2.4.4), stats (>= 4.4.1), tidyr (>= 1.3.1), utils (>= 4.4.1), rlang (>= 1.1.0), magrittr (>= 2.0.3), future.apply, future

Config/testthat/edition 3

NeedsCompilation no

Author Francois Bassac [aut, cre]

Maintainer Francois Bassac <fr.bassac@gmail.com>

Repository CRAN

Date/Publication 2026-05-04 19:20:02 UTC

Contents

assemble_basis_metric . . . 4
assert_funcPLS_inputs . . . 5
assert_multivariate_naivePLS_inputs . . . 6
assert_multivariate_smoothPLS_inputs . . . 7
beta_1_real_func . . . 8
beta_2_real_func . . . 8
beta_3_real_func . . . 9
beta_4_real_func . . . 10
beta_5_real_func . . . 11
beta_6_real_func . . . 11
beta_7_real_func . . . 12
beta_list_generation . . . 13
block_diag . . . 14
build_df_per_state . . . 14
build_new_data_list . . . 15
build_reg_curve_spls . . . 16
build_spls_functions . . . 17
build_u_ki_list . . . 18
cat_data_to_indicator . . . 18
convert_to_wide_format . . . 19
create_bspline_basis . . . 20
determine_curve_name . . . 21
determine_next_state . . . 21
evaluate_curves_distances . . . 22
evaluate_gamma_ij . . . 23
evaluate_id_func_integral . . . 23
evaluate_id_func_integral_deprecated . . . 24
evaluate_lambda . . . 25
evaluate_lambda_CFD . . . 26
evaluate_lambda_CFD_para_v1 . . . 27
evaluate_lambda_SFD . . . 28
evaluate_lambda_SFD_para_v1 . . . 29
evaluate_metric . . . 30
evaluate_reg_curve_SPLS_uni . . . 31
evaluate_results . . . 32
evaluate_variance_explained . . . 32
evaluate_V_i_function . . . 33

eval_max_min_y . . . 34
from_basis_to_fdlist . . . 34
from_fd_to_func . . . 35
funcPLS . . . 36
funcPLS_predict . . . 37
generate_probabilities . . . 38
generate_X_df . . . 39
generate_X_df_CFD . . . 40
generate_X_df_multistates . . . 41
generate_X_df_SFD . . . 42
generate_X_df_SFD_data . . . 42
generate_X_df_test . . . 43
generate_Y_df . . . 44
generate_Y_df_CFD . . . 46
generate_Y_df_SFD . . . 47
gram_schmidt_orthonormalize . . . 48
help_smoothPLS . . . 49
initial_state_determination . . . 49
is_orthogonal . . . 50
is_orthonormal . . . 50
lambda_determination . . . 51
mae_values . . . 51
multivariate_alpha_building . . . 52
multivariate_assemble_basis_metric . . . 53
naivePLS . . . 54
naivePLS_formatting . . . 55
naivePLS_predict . . . 56
number_of_test_id . . . 57
obj_list_creation . . . 57
orthonormalize_basis_list . . . 58
plot_CFD_individuals . . . 59
plot_fd_list . . . 60
plot_model_metrics_base . . . 60
plot_real_and_smoothed_data_ind . . . 61
press_model . . . 62
press_values . . . 63
p_w_building . . . 64
regularize_time_series . . . 65
regularize_time_series_CFD . . . 66
regularize_time_series_SFD . . . 66
reg_curve_funcPLS_evaluation . . . 67
remove_duplicate_states . . . 68
r_squared_values . . . 69
select_from_fd_list . . . 70
smoothPLS . . . 70
smoothPLS_CFD_predict . . . 72
smoothPLS_CFD_predict_para_v1 . . . 73
smoothPLS_predict . . . 74

smoothPLS_predict_uni 75
smoothPLS_SFD_predict 76
smoothPLS_SFD_predict_para_v1 77
split_in_state_df 78
state_indicator 79
state_indicator_old 80
test_basis_properties 81
transfer_probabilities 81
transition_matrix 82
univariate_alpha_building 83

Index **84**

`assemble_basis_metric` *assemble_basis_metric*

Description

This function assemble the metrics of all the basis of the basis list. This function only assemble the needed basis, especially if length(curve_to_keep) != N_states

Usage

```
assemble_basis_metric(basis_list, curves_to_keep = NULL)
```

Arguments

`basis_list`	a list of basis fd object
`curves_to_keep`	a list of the states curves to keep

Value

a matrix of the metric to consider

Author(s)

Francois Bassac

Examples

```
basis1 = fda::create.bspline.basis(c(0,100), nbasis=10, norder=4)
basis2 = fda::create.bspline.basis(c(0,100), nbasis=15, norder=1)
basis3 = fda::create.fourier.basis(c(0,100), nbasis=7)
assemble_basis_metric(list(basis1, basis2, basis3), list(1,2,4))
assemble_basis_metric(list(basis1, basis2, basis3), list(1,2))
```

`assert_funcPLS_inputs` *assert_funcPLS_inputs*

Description

This function checks the integrity of the input for funcPLS. It returns a list of (basis_list, regul_time_list, curve_type_list, id_col_list, time_col_list)

Usage

```
assert_funcPLS_inputs(
  df_list,
  Y,
  basis_obj,
  regul_time_obj,
  curve_type_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time"
)
```

Arguments

`df_list`	a list of dataframes (id, time, value_or_state)
`Y`	a numeric vector of the response
`basis_obj`	a list of basis object or a basis object
`regul_time_obj`	a vector of time regularization values or a list of vectors
`curve_type_obj`	a character "cat" or 'num' or a list of those values
`id_col_obj`	a character of the id column for all the curves or a list of id column character
`time_col_obj`	a character of the time column for all the curves or a list of time column character

Value

a list of (basis_list, regul_time_list, curve_type_list, id_col_list, time_col_list)

Author(s)

Francois Bassac

assert_multivariate_naivePLS_inputs
assert_multivariate_naivePLS_inputs

Description

This function checks the input of naivePLS function.

Usage

```
assert_multivariate_naivePLS_inputs(
  df_list,
  Y,
  regul_time_obj = NULL,
  curve_type_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time"
)
```

Arguments

df_list	a list of dataframe (id, time, value_or_state)
Y	a numeric vector
regul_time_obj	a list of time regularisation values
curve_type_obj	a list of curve type 'cat' or 'num'
id_col_obj	a list of the names of the id columns
time_col_obj	a list of the names of the time columns

Value

a list

Author(s)

Francois Bassac

assert_multivariate_smoothPLS_inputs
assert_multivariate_smoothPLS_inputs

Description

This function checks the integrity of the input for multivariate_fpls. It returns a list of (basis_list, regul_time_list, curve_type_list, id_col_list, time_col_list)

Usage

```
assert_multivariate_smoothPLS_inputs(
  df_list,
  Y,
  basis_obj,
  regul_time_obj = NULL,
  curve_type_obj = NULL,
  orth_obj = list(TRUE),
  id_col_obj = "id",
  time_col_obj = "time"
)
```

Arguments

df_list	a list of dataframes (id, time, value_or_state)
Y	a numeric vector of the response
basis_obj	a list of basis object or a basis object
regul_time_obj	a vector of time regularization values or a list of vectors
curve_type_obj	a character "cat" or 'num' or a list of those values
orth_obj	a boolean, a list or a vector of boolean to orthonormalize or not a basis
id_col_obj	a character of the id column for all the curves or a list of id column character
time_col_obj	a character of the time column for all the curves or a list of time column character

Value

a list of (basis_list, regul_time_list, curve_type_list, orth_list, id_col_list, time_col_list)*

Author(s)

Francois Bassac

beta_1_real_func *beta_1_real_func*

Description

beta_1_real_func

Usage

```
beta_1_real_func(t, end_time = 100, drop = NULL)
```

Arguments

t	evaluation time
end_time	end time; default 100
drop	particular point of the curve, default NULL

Value

a value

Author(s)

Francois Bassac

Examples

```
beta_1_real_func(0)
beta_1_real_func(10)
beta_1_real_func(10:90)
plot(x=0:100, y=beta_1_real_func(0:100, 100), type='l', main="Beta_1")
```

beta_2_real_func *beta_2_real_func*

Description

beta_2_real_func

Usage

```
beta_2_real_func(t, end_time = 100, drop = 3 * 100/5)
```

Arguments

t	evaluation time
end_time	end time; default 100
drop	particular point of the curve, default 3*100/5

Value

a value

Author(s)

Francois Bassac

Examples

```
beta_2_real_func(0)
beta_2_real_func(10)
beta_2_real_func(10:90)
plot(x=0:100, y=beta_2_real_func(0:100, 100), type='l', main="Beta_2")
```

---

beta_3_real_func *beta_3_real_func*

Description

beta_3_real_func

Usage

```
beta_3_real_func(t, end_time = 100, drop = 27 * 100/100)
```

Arguments

t	evaluation time
end_time	end time; default 100
drop	particular point of the curve, default 27

Value

a value

Author(s)

Francois Bassac

Examples

```
beta_3_real_func(0)
beta_3_real_func(10)
beta_3_real_func(10:90)
plot(x=0:100, y=beta_3_real_func(0:100, 100), type='l', main="Beta_3")
```

`beta_4_real_func` *beta_4_real_func*

Description

beta_4_real_func

Usage

```
beta_4_real_func(t, end_time = 100, drop = NULL)
```

Arguments

`t`	evaluation time
`end_time`	end time; default 100
`drop`	particular point of the curve, default NULL

Value

a value

Author(s)

Francois Bassac

Examples

```
beta_4_real_func(0)
beta_4_real_func(10)
beta_4_real_func(10:90)
plot(x=0:100, y=beta_4_real_func(0:100, 100), type='l', main="Beta_4")
```

beta_5_real_func *beta_5_real_func*

Description

beta_5_real_func

Usage

```
beta_5_real_func(t, end_time = 100, drop = 3 * 100/5)
```

Arguments

t	evaluation time
end_time	end time; default 100
drop	particular point of the curve, default 3*100/5

Value

a value

Author(s)

Francois Bassac

Examples

```
beta_5_real_func(0)
beta_5_real_func(10)
beta_5_real_func(10:90)
plot(x=0:100, y=beta_5_real_func(0:100, 100), type='l', main="Beta_5")
```

beta_6_real_func *beta_6_real_func*

Description

Constant function = 1 Can adjust the constant value by drop input

Usage

```
beta_6_real_func(t, end_time = 100, drop = 1)
```

Arguments

t	evaluation time
end_time	end time; default 100
drop	particular point of the curve, default 1

Value

a value

Author(s)

Francois Bassac

Examples

```
beta_5_real_func(0)
beta_5_real_func(10)
beta_5_real_func(10:90)
plot(x=0:100, y=beta_5_real_func(0:100, 100), type='l', main="Beta_5")
```

---

beta_7_real_func *beta_7_real_func*

Description

Triangular function with angle at (x=drop, y=drop) with slope of 1 and -1

Usage

```
beta_7_real_func(t, end_time = 100, drop = 3 * end_time/5)
```

Arguments

t	evaluation time
end_time	end time; default 100
drop	particular point of the curve, default 3*end_time/5

Value

a value

Author(s)

Francois Bassac

Examples

```
beta_5_real_func(0)
beta_5_real_func(10)
beta_5_real_func(10:90)
plot(x=0:100, y=beta_5_real_func(0:100, 100), type='l', main="Beta_5")
```

`beta_list_generation` *beta_list_generation*

Description

beta_list_generation

Usage

```
beta_list_generation(N_states = 3)
```

Arguments

`N_states`	a int of the number of states wanted, default 3

Value

a list of functions

Author(s)

Francois Bassac

Examples

```
beta_list = beta_list_generation()
beta_list_2 = beta_list_generation(6)
```

block_diag *block_diag*

Description

returns a matrix whose blocks are A and B. returns : (A, 0) (0, B)

Usage

```
block_diag(A, B)
```

Arguments

A	a matrix
B	a matrix

Value

a matrix

Author(s)

Francois Bassac

Examples

```
A = matrix(c(1, 2, 3, 4), 2)
B = matrix(c(5, 6, 7, 8,9, 10), 2)
C = block_diag(A, B)
```

build_df_per_state *build_df_per_state*

Description

build_df_per_state

Usage

```
build_df_per_state(data_list, id_col = "id", time_col = "time")
```

Arguments

data_list	a list containing the dataframe of the indicator function of each state.
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'

Value

a list of the ordered states in the indicator function form.

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas, transition_df)

si_df = state_indicator(df, id_col='id',
time_col='time')

split_df = split_in_state_df(si_df, id_col='id', time_col='time')
```

---

build_new_data_list *build_new_data_list*

Description

This function preprocess the different input in order to format them to the right number of curve. Warning the "right" number of curve take into account the number of different states for the CFDs.

Usage

```
build_new_data_list(
  df_list,
  N_curves,
  orth_basis_list = NULL,
  basis_list = NULL,
  curve_type_list,
  id_col_list,
  time_col_list,
  regul_time_list
)
```

Arguments

df_list	a list of dataframes (id, time, value_or_state)
N_curves	a integer, the number of curves

`orth_basis_list`
: a list of orthogonalized basis fd list

`basis_list`
: a list of basis fd functions

`curve_type_list`
: a list of the curve type of each curve

`id_col_list`
: a list of the id column name for each curve

`time_col_list`
: a list of the time column name for each curve

`regul_time_list`
: a list of the time regularization vector for each curve

Value

a list

Author(s)

Francois Bassac

`build_reg_curve_spls` *build_reg_curve_spls*

Description

This function builds the smooth PLS regression functions.

Usage

```
build_reg_curve_spls(
  plsr_model,
  curves_names_list,
  v_i_list,
  nb_comp_pls_opt = NULL,
  print_steps = TRUE
)
```

Arguments

`plsr_model`
: a pls model

`curves_names_list`
: a list of the names of the different curves

`v_i_list`
: a list of the v functions

`nb_comp_pls_opt`
: a integer, the number of component to take into account. if null (default) all the components are used

`print_steps`
: a boolean to print steps

Value

a list of fd object

Author(s)

Francois Bassac

`build_spls_functions` *build_spls_functions*

Description

This function build some intermediate functions of the smooth pls algorithm. It also evaluates the different coefficients gamma_ij

Usage

```
build_spls_functions(
  curves_names_list,
  new_basis_list,
  new_orth_basis_list,
  d_i,
  u_i
)
```

Arguments

`curves_names_list`	a list of the names of the different curves
`new_basis_list`	a list of the initial basis fd object for all curves
`new_orth_basis_list`	a list of the orthogonalized basis as fd list for all curves
`d_i`	the pls coefficient such as X = d_i t_i : plsr_model$loadings
`u_i`	the pls coefficient such as t_i = X u_i : plsr_model$loading.weights

Value

a list Lambda = sum d_i t_i

Author(s)

Francois Bassac

build_u_ki_list *build_u_ki_list*

Description

build_u_ki_list

Usage

```
build_u_ki_list(N_states, nbComp, ms_pls_models)
```

Arguments

N_states	a integer, number of different states
nbComp	a integer, max number of components
ms_pls_models	a list of the intermediate pls models to evaluate the real multi-stats pls components.

Value

a list of the u_i^k

Author(s)

Francois Bassac

cat_data_to_indicator *cat_data_to_indicator*

Description

This function apply all functions to go from a categorical functional data with different states to a list of one dataframe per state indicatrice (in the ascending order) whose duplicated states where removed.

Usage

```
cat_data_to_indicator(data, id_col = "id", time_col = "time")
```

Arguments

data	a multistates dataframe ('id', 'time', 'states')
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'

Value

a list of the ordered states in the indicator function form.

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas, transition_df)
df_list = cat_data_to_indicator(df)
```

`convert_to_wide_format` *convert_to_wide_format*

Description

This function takes the df_new output from regularize_time_series and convert it into another format

Usage

```
convert_to_wide_format(df_new, id_col = "id", time_col = "time")
```

Arguments

`df_new`	a regularized dataframe
`id_col`	col_name of df_new for the id
`time_col`	col_name of df_new for the id

Value

the dataframe in wide format

Author(s)

Francois Bassac

Examples

```
id_df = data.frame(id=rep(1,5), time=seq(0, 40, 10), state=c(0, 1, 1, 0, 1))
id_df_new = regularize_time_series(id_df, time_seq = seq(0, 40, 2), curve_type = 'cat')
convert_to_wide_format(id_df_new)
```

---

create_bspline_basis *create_bspline_basis*

Description

create_bspline_basis

Usage

```
create_bspline_basis(start, end, nbasis = 10, norder = 4)
```

Arguments

start	start time
end	end time
nbasis	number of basis functions, default 10
norder	order of the basis function, default cubic splines 4

Value

a basis fd object

Author(s)

Francois Bassac

Examples

```
b0 = create_bspline_basis(0, 10, 10, 4)
plot(b0)

b1 = create_bspline_basis(0, 10, 10, 2)
plot(b1)

b2 = create_bspline_basis(0, 10, 10, 1)
plot(b1)
```

determine_curve_name *determine_curve_name*

Description

This function determines the curves names bases on its place in the data and its states if it is a categorical functional data.

Usage

```
determine_curve_name(curve_number, curve_type = NULL, states_names = NULL)
```

Arguments

curve_number	a int, the place of the curve in the data
curve_type	a character, 'cat' or 'num'
states_names	a character or list of character with the states of the CFD

Value

a vector of names.

Author(s)

Francois Bassac

determine_next_state *determine_next_state*

Description

This function determines the next state base on the current state and the transition matrix.

Usage

```
determine_next_state(current_state, transition_df)
```

Arguments

current_state	a value of the current state
transition_df	a dataframe of the transition matrix

Value

a value for the next state

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)
determine_next_state(1, transition_df)
```

`evaluate_curves_distances`
evaluate_curves_distances

Description

Either for R func or fd function, this function evaluates the distance between the real curve and each curve fun or fd which are in the func_fd_list.

Usage

```
evaluate_curves_distances(real_f, regul_time, fun_fd_list = NULL)
```

Arguments

`real_f`	a fun of fd function, base function to compare
`regul_time`	a vector of time regularization values
`fun_fd_list`	a list of fun or fd functions or a fun or a fd function

Value

No return value, called for side effects (prints distances to the console).

Author(s)

Francois Bassac

evaluate_gamma_ij *evaluate_gamma_ij*

Description

This function evaluates int_0^T w_i_fd p_j_t dt for all j return a list, the element i length is (i-1)

Usage

```
evaluate_gamma_ij(w_i_list, p_i_list)
```

Arguments

w_i_list	a list of fd functions w_i(t)
p_i_list	a list of fd functions p_i(t)

Value

a list of list of the gamma_ij values

Author(s)

Francois Bassac

evaluate_id_func_integral *evaluate_id_func_integral*

Description

Evaluates the integral $\int_{\tau_i} f(t)dt$ where τ_i are the active intervals of a categorical functional data (states 0 or 1).

Usage

```
evaluate_id_func_integral(
  id_df,
  func,
  id_col = "id",
  time_col = "time",
  rel_tol = .Machine$double.eps^0.5,
  subdivisions = 1000L,
  ...
)
```

Arguments

id_df	Dataframe for a single individual with at least columns (id, time, state).
func	The R function to integrate.
id_col	Character, name of the id column, default 'id'.
time_col	Character, name of the time column, default 'time'.
rel_tol	Relative tolerance for stats::integrate, default 1e-8.
subdivisions	Max number of subdivisions for integrate, default 100.
...	Additional arguments (ignored to prevent passing unused params to func).

Value

A dataframe with the id and the calculated integral value.

---

evaluate_id_func_integral_deprecated *evaluate_id_func_integral*

Description

This function evaluate the integral for a state (0, 1) functional data : int(X(t) func(t))dt. This function works ONLY for a one state CFD!

Usage

```
evaluate_id_func_integral_deprecated(
  id_df,
  func,
  mode = 1,
  id_col = "id",
  time_col = "time",
  nb_pt = 10,
  subdivisions = 100
)
```

Arguments

id_df	a single id dataframe of at least named columns (id, time)
func	the function to integrate
mode	select the integration mode 1 for R function integrate, 2 for pracma::trapz. default value : 1
id_col	col_name of df for the id
time_col	col_name of df for the time
nb_pt	number of points for the integration, default value : 10
subdivisions	default parameter of R function integrate; default value : 100

Value

a dataframe with the id and the integral value.

Author(s)

Francois Bassac

Examples

```
id_df = data.frame(id=rep(1,5), time=seq(0, 40, 10), state=c(0, 1, 1, 0, 1))
evaluate_id_func_integral(id_df, function(t){t})
```

`evaluate_lambda` *evaluate_lambda*

Description

This function evaluates the Lambda matrix such as per column : Lambda_i = int_0^T X(t) phi_i(t) dt. The curve_type input is important function of the type of data you work with 'cat' for Categorical Functional Data 'num' for Scalar Functional Data

Usage

```
evaluate_lambda(
  df,
  basis,
  curve_type = NULL,
  int_mode = 1,
  id_col = "id",
  time_col = "time",
  nb_pt = 10,
  subdivisions = 100,
  regul_time = seq(basis$rangeval[1], basis$rangeval[2], 1),
  parallel = TRUE
)
```

Arguments

`df`	dataframe X(t)
`basis`	basis fd object
`curve_type`	a character, 'cat' for Categorical FD, 'num' for Scalar FD
`int_mode`	integration mode, 1 for integrate, 2 for pracma::trapz
`id_col`	a character for the id column, default 'id'
`time_col`	a character for the time column, default 'time'

nb_pt	number of points for the integration, default value : 10
subdivisions	default parameter of R function integrate; default value : 100
regul_time	regul_time a vector of time regularization values default basis rangeval per 1
parallel	a boolean to use parallelization, default TRUE

Value

a matrix $\Lambda_i = \int_0^T X(t)\phi_i(t)dt$

Author(s)

Francois Bassac

Examples

```
df = generate_X_df(nind=100, start=0, end=100, curve_type = 'cat',
lambda_0=0.2, lambda_1=0.1, prob_start=0.5)
basis = create_bspline_basis(0, 100, 10, 4)
Lambda = evaluate_lambda(df, basis, curve_type = 'cat')


df = generate_X_df(nind=100, start=0, end=100, curve_type = 'num')
basis = create_bspline_basis(0, 100, 10, 4)
Lambda = evaluate_lambda(df, basis, curve_type = 'num', int_mode = 2)
```

---

evaluate_lambda_CFD *evaluate_lambda_CFD*

Description

This function evaluates the Lambda matrix Lambda_ij = int X_j(t) phi_i(t) dt using parallel processing with Chunking.

Usage

```
evaluate_lambda_CFD(
  df,
  basis,
  int_mode = 1,
  id_col = "id",
  time_col = "time",
  nb_pt = 10,
  subdivisions = 100,
  parallel = TRUE
)
```

Arguments

df	dataframe X(t)
basis	basis fd object or a list of fd functions
int_mode	integration mode, 1 for integrate, 2 for pracma::trapz
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'
nb_pt	number of points for the integration, default value : 10
subdivisions	default parameter of R function integrate; default value : 100
parallel	boolean, if TRUE uses max(nb_cores - 2, 1) cores, default TRUE.

Value

a matrix of dimension nbasis columns and nind rows

Author(s)

Francois Bassac

`evaluate_lambda_CFD_para_v1` *evaluate_lambda_CFD_para_v1*

Description

This function evaluates the Lambda matrix Lambda_ij = int X_j(t) phi_i(t) dt using parallel processing.

Usage

```
evaluate_lambda_CFD_para_v1(
  df,
  basis,
  int_mode = 1,
  id_col = "id",
  time_col = "time",
  nb_pt = 10,
  subdivisions = 100,
  parallel = TRUE
)
```

Arguments

df	dataframe X(t)
basis	basis fd object or a list of fd functions
int_mode	integration mode, 1 for integrate, 2 for pracma::trapz
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'
nb_pt	number of points for the integration, default value : 10
subdivisions	default parameter of R function integrate; default value : 100
parallel	a boolean to enable parallel processing, default TRUE

Value

a matrix of dimension nbasis columns and nind rows

Author(s)

Francois Bassac

---

evaluate_lambda_SFD *evaluate_lambda_SFD*

Description

This function evaluates the Lambda matrix Lambda_ij = int X_j(t) phi_i(t) dt for step > 1 using parallel chunking.

Usage

```
evaluate_lambda_SFD(
  df,
  basis,
  regul_time = seq(basis$rangeval[1], basis$rangeval[2], 1),
  int_mode = 1,
  id_col = "id",
  time_col = "time",
  subdivisions = 100,
  parallel = TRUE
)
```

Arguments

df	dataframe X(t)
basis	basis fd object
regul_time	a vector of time regularization values default basis rangeval per 1
int_mode	int, integration mode, 1 for integrate, 2 for pracma::trapz
id_col	default name of the id column
time_col	default name of the time column
subdivisions	default parameter of R function integrate; default value : 100
parallel	boolean, if TRUE uses parallel processing. Default TRUE.

Value

a matrix of dimension nbasis columns and nind rows

Author(s)

Francois Bassac

`evaluate_lambda_SFD_para_v1` *evaluate_lambda_SFD_para_v1*

Description

This function evaluates the Lambda matrix Lambda_ij = int X_j(t) phi_i(t) dt for step > 1

Usage

```
evaluate_lambda_SFD_para_v1(
  df,
  basis,
  regul_time = seq(basis$rangeval[1], basis$rangeval[2], 1),
  int_mode = 1,
  id_col = "id",
  time_col = "time",
  subdivisions = 100,
  parallel = TRUE
)
```

Arguments

df	dataframe X(t)
basis	basis fd object
regul_time	a vector of time regularization values default basis rangeval per 1
int_mode	int, integration mode, 1 for integrate, 2 for pracma::trapz
id_col	default name of the id column
time_col	default name of the time column
subdivisions	default parameter of R function integrate; default value : 100
parallel	a boolean to enable parallel processing, default TRUE

Value

a matrix of dimension nbasis columns and nind rows

Author(s)

Francois Bassac

---

evaluate_metric *evaluate_metric*

Description

This function evaluates the metric of a certain basis. The metric is the inprod of the basis functions.

Usage

```
evaluate_metric(basis)
```

Arguments

basis	basis to evaluate the metric

Value

a matrix of dimension nbasis X nbasis

Author(s)

Francois Bassac

Examples

```
basis = create_bspline_basis(start=0, end=10, nbasis=10, norder=4)
metric = evaluate_metric(basis)

basis1 = create_bspline_basis(start=0, end=20, nbasis=10, norder=1)
metric1 = evaluate_metric(basis1)
```

---

evaluate_reg_curve_SPLS_uni

evaluate_reg_curve_SPLS_uni

Description

This functions evaluates the regression function such as : $Y = \int_0^T delta(t)X(t)dt$.

Usage

```
evaluate_reg_curve_SPLS_uni(plsr_model, v_i_list, nb_comp = NULL)
```

Arguments

plsr_model	a model from pls package
v_i_list	a list of fd functions v_i(t) : $t_i = \int_0^T v_i(t)X(t)dt$
nb_comp	a value, number of components to take into account, default NULL if NULL, then delta(t) will take every components available.

Value

a fd function

Author(s)

Francois Bassac

`evaluate_results` *evaluate_results This function evaluates the PRESS, RMSE, MAE, R2 and the % of variance between Y and Y_hat*

Description

evaluate_results This function evaluates the PRESS, RMSE, MAE, R2 and the % of variance between Y and Y_hat

Usage

```
evaluate_results(Y, Y_hat)
```

Arguments

Y	a vector of real values
Y_hat	a vector of modeled values

Value

a dataframe

Author(s)

Francois Bassac

Examples

```
evaluate_results(c(1,2,3,4,5), c(0.9, 2.2, 4, 5.5, 5))
```

`evaluate_variance_explained` *evaluate_variance_explained*

Description

This function return the % of variance explained by Y_hat comparing to Y.

Usage

```
evaluate_variance_explained(Y, Y_hat)
```

Arguments

Y	a reference value
Y_hat	a modeled value Y_hat = model(X)

Value

a value in %

Author(s)

Francois Bassac

Examples

```
evaluate_variance_explained(c(1,2,3,4,5,6,7,8,9,10),
c(0.9, 1.1, 1.9, 2.4, 5.3, 6.01, 7,45, 9.12, 9.04, 11.6))
```

---

evaluate_V_i_function *evaluate_V_i_function*

Description

This function evaluates the different functions v_i_t bases on gamma_ij and the functions w_i_fd. Recursive : $v_i = W_i - \sum_{j=1}^{i-1} gamma_i j * v_j$

Usage

```
evaluate_V_i_function(w_i_list, gamma_ij_list)
```

Arguments

w_i_list	a list of fd functions w_i(t)
gamma_ij_list	a list of list of gamma_ij values

Value

a list of fd functions

Author(s)

Francois Bassac

eval_max_min_y *eval_max_min_y*

Description

This function returns the min and max values of a list of functions and fd objects on regul_time.

Usage

```
eval_max_min_y(f_list, regul_time)
```

Arguments

f_list	a list of functions and fd objects
regul_time	a vector of time evaluation points.

Value

a vector

Author(s)

Francois Bassac

from_basis_to_fdlist *from_basis_to_fdlist*

Description

This function transform if necessary the input basis into a list of fd functions. If basis is a basis object from fda, the output fd_list is the list of the different basis functions as fd functions. If basis is already a list of fd functions, nothing changes.

Usage

```
from_basis_to_fdlist(basis)
```

Arguments

basis	a basis fd object or a list of fd functions.

Value

a list of fd functions

Author(s)

Francois Bassac

Examples

```
basis = create_bspline_basis(start = 0, end = 10, nbasis = 5, norder = 4)
plot(basis)

basis_list = from_basis_to_fdlist(basis)

plot(basis_list[[1]], col = 1)
for(i in 2:length(basis_list)){
  plot(basis_list[[i]], add=TRUE, col = i)
}

basis_list_2 = from_basis_to_fdlist(basis_list)
```

---

from_fd_to_func *from_fd_to_func*

Description

This function transform a fd object into a function. It require either the fd object OR the coefficient and the basis object.

Usage

```
from_fd_to_func(fd_obj = NULL, coef = NULL, basisobj = NULL)
```

Arguments

fd_obj	a fd object to transform into a function
coef	the coefficient of an fd object to transform into a function
basisobj	the basis object of an fd object to transform into a function

Value

a function

Author(s)

Francois Bassac

Examples

```
basis = create_bspline_basis(0, 100, 10, 4)
coef = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
func_from_fd = from_fd_to_func(coef = coef, basis = basis)
```

funcPLS *funcPLS*

Description

This function performs the Multivariate Functional PLS as a matrix problem.

Usage

```
funcPLS(
  df_list,
  Y,
  basis_obj,
  regul_time_obj,
  curve_type_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time",
  print_steps = FALSE,
  plot_rmsep = TRUE,
  print_nbComp = TRUE,
  plot_reg_curves = FALSE,
  jackknife = TRUE,
  validation = "LOO"
)
```

Arguments

df_list	a list of dataframes (id, time, value_or_state)
Y	a numeric vector of the response
basis_obj	a basis fd obj or a list of basis fd obj. If basis fd obj, the same basis is used for all the curves
regul_time_obj	a vector of time regularization values or a list of vectors
curve_type_obj	a character "cat" or 'num' or a list of those values
id_col_obj	a character of the id column for all the curves or a list of id column character
time_col_obj	a character of the time column for all the curves or a list of time column character
print_steps	a boolean to cat the current step
plot_rmsep	a boolean to plot the plsr RMSEP
print_nbComp	a boolean to cat the optimal number of components
plot_reg_curves	a boolean to directly plot the beta regression curves
jackknife	a plsr input, default = TRUE
validation	a plsr input, default = 'LOO'

Value

a list ("curve_names", "alphas", "metric", "root_metric", "trans_alphas", "mfpls_mfd", "nb_comp_pls_opt", "beta_0", "beta_pls_list")

Author(s)

Francois Bassac

`funcPLS_predict` *funcPLS_predict*

Description

This function performs the prediction on a df_predict_ms using the delta_list $\hat{(Y)} = \delta_0 + \sum_{i=1}^{K} \int_0^T \delta_i(t) X_i(t) dt$.

Usage

```
funcPLS_predict(
  df_predict_list,
  delta_list,
  curve_type_obj = NULL,
  regul_time_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time",
  int_mode = 1,
  nb_pt = 10,
  subdivisions = 100,
  parallel = TRUE
)
```

Arguments

`df_predict_list`	a list of dataframe (id, time, state_or_value)
`delta_list`	a list of regression objects (Intercept, fd, etc)
`curve_type_obj`	a list of the curves types 'cat' or 'num'
`regul_time_obj`	a list of time regularization values
`id_col_obj`	a list of characters of the names of the id columns
`time_col_obj`	a list of characters of the names of the time columns
`int_mode`	a integer for integration mode, 1 for integrate, 2 for pracma::trapz, default 1
`nb_pt`	a integer, the number of intermediate points for integration mode 2, default 10
`subdivisions`	a integer, the number of subdivisions for integration mode 1, default 100
`parallel`	a boolean to use parallelization, default TRUE

Value

a numeric vector

Author(s)

Francois Bassac

`generate_probabilities` *generate_probabilities*

Description

This function generates probabilities whose sum is 1.

Usage

```
generate_probabilities(N_proba)
```

Arguments

`N_proba`	a int the number of values requested

Value

a vector of the probabilities

Author(s)

Francois Bassac

Examples

```
generate_probabilities(3)
generate_probabilities(5)
```

generate_X_df *generate_X_df*

Description

This function generate synthetic data of nind X(t). For 'cat' curve_type it is in state 0 or 1 by two different exponential laws. For curve_type = 'num' it is noised cosine.

Usage

```
generate_X_df(
  nind = 100,
  start = 0,
  end = 100,
  curve_type = "cat",
  lambda_0 = 0.2,
  lambda_1 = 0.1,
  prob_start = 0.5,
  noise_sd = 0.1,
  seed = 123
)
```

Arguments

nind	number of individuals, default 100
start	first time, default 0
end	last time, default 100
curve_type	character, type of wanted synthetic data, default 'cat', need to be \'cat\' or \'num\'.
lambda_0	lambda parameter for exponential law for state 0, default 0.2
lambda_1	lambda parameter for exponential law for state 1, default 0.1
prob_start	Start state 1 probability, binomial law, default 0.5
noise_sd	noise added to the signal, default 0.1
seed	seed for reproducibility, default 123

Value

the dataframe of the individuals

Author(s)

Francois Bassac

Examples

```
generate_X_df()
```

generate_X_df_CFD *generate_X_df_CFD*

Description

This function generate synthetic data of nind X(t) in state 0 or 1 by two different exponential laws.

Usage

```
generate_X_df_CFD(
  nind = 500,
  start = 0,
  end = 100,
  lambda_0 = 0.2,
  lambda_1 = 0.1,
  prob_start = 0.5,
  seed = 123
)
```

Arguments

nind	number of individuals, default 500
start	first time, default 0
end	last time, default 100
lambda_0	lambda parameter for exponential law for state 0, default 0.2
lambda_1	lambda parameter for exponential law for state 1, default 0.1
prob_start	Start state 1 probability, binomial law, default 0.5
seed	a integer, random seed 123

Value

the dataframe of the individuals

Author(s)

Francois Bassac

Examples

```
generate_X_df_CFD()
generate_X_df_CFD(10, 13, 60, 0.21, 0.13, 0.7)
```

generate_X_df_multistates *generate_X_df_multistates*

Description

This function generates a multistates CFD.

Usage

```
generate_X_df_multistates(
  nind = 100,
  N_states = 3,
  start = 0,
  end = 100,
  lambdas,
  transition_df,
  seed = 123
)
```

Arguments

nind	a int of the number of the individuals, default 100
N_states	a int of the number of states wanted, default 3
start	a value of starting time, default 0
end	a value of ending time, default 100
lambdas	a vector of N_states lambda values from lambda_determination(N_states)
transition_df	a dataframe with the transition matrix from transfer_probabilities(N_states)
seed	a integer, random seed

Value

a dataframe of a multistates Categorical Functional Data

Author(s)

Francois Bassac

Examples

```
N_states = 4
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas,  transition_df)
```

generate_X_df_SFD *generate_X_df_SFD*

Description

This function generate synthetic data of nind X which are noised cosines.

Usage

```
generate_X_df_SFD(nind = 100, start = 0, end = 100, noise_sd = 0.1, seed = 123)
```

Arguments

nind	number of individuals, default 500
start	first time, default 0
end	last time, default 100
noise_sd	noise added to the signal, default 0.1
seed	seed for reproducibility, default 123

Value

a dataframe

Author(s)

Francois Bassac

Examples

```
generate_X_df_SFD(nind = 100, start = 0, end = 100,
noise_sd = 0.1, seed = 123)
```

generate_X_df_SFD_data *generate_X_df_SFD_data*

Description

This function generate test data of nind X which are noised cosines.

Usage

```
generate_X_df_SFD_data(
  nind = 100,
  start = 0,
  end = 100,
  noise_sd = 0.1,
  seed = 123
)
```

Arguments

nind	number of individuals, default 500
start	first time, default 0
end	last time, default 100
noise_sd	noise added to the signal, default 0.1
seed	seed for reproducibility, default 123

Value

a dataframe

Author(s)

Francois Bassac

Examples

```
generate_X_df_SFD(nind = 100, start = 0, end = 100,
noise_sd = 0.1, seed = 123)
```

---

generate_X_df_test *generate_X_df_test*

Description

generate_X_df_test

Usage

```
generate_X_df_test(
  TTRatio = 0.2,
  nind = 500,
  start = 0,
  end = 100,
  curve_type = "cat",
```

```
  lambda_0 = 0.2,
  lambda_1 = 0.1,
  prob_start = 0.5,
  seed = 123
)
```

Arguments

TTRatio	Train Test ratio, default 0.2
nind	number of individuals, default 500
start	first time, default 0
end	last time, default 100
curve_type	character, type of wanted synthetic data, default 'cat', need to be \'cat\' or \'num\'.
lambda_0	lambda parameter for exponential law for state 0, default 0.2
lambda_1	lambda parameter for exponential law for state 1, default 0.1
prob_start	Start state 1 probability, binomial law, default 0.5
seed	a integer, random seed 123

Value

a dataframe of the test data

Author(s)

Francois Bassac

Examples

```
generate_X_df_test()
generate_X_df_test(TTRatio = 0.4, nind=8, start=0, end=10, curve_type = 'num')
```

---

generate_Y_df *generate_Y_df*

Description

This function generates Y_df bases on df, beta_func with the following link Y = beta_0 + int(X(t)*beta(t))dt It generates also the noised values of Y. Here the NotS_ratio is the Noise over total Signal ratio meaning that a value of 0.2 means that the noise represents 20% of the TOTAL variance.

Usage

```
generate_Y_df(
  df,
  curve_type = NULL,
  beta_real_func_or_list,
  beta_0_real = 5.4321,
  NotS_ratio = 0.2,
  seed = 123,
  id_col = "id",
  time_col = "time",
  int_mode = 1,
  nb_pt = 10,
  subdivisions = 100,
  parallel = FALSE
)
```

Arguments

df	the X(t) dataframe to evaluate Y on
curve_type	a character, the type of data, default NULL, need to be \'cat\' or \'num\'.
beta_real_func_or_list	a function or a list of functions beta(t), function used for the Y evaluation
beta_0_real	the intercept, default 5.4321
NotS_ratio	the Noise over total Signal ratio, default 0.2
seed	a integer value for the seed to be reproducible 123
id_col	a character of the id column, default 'id'
time_col	a character of the time column, default 'time'
int_mode	integration mode, 1 for integrate, 2 for pracma::trapz
nb_pt	number of points for the integration, default value : 10
subdivisions	default parameter of R function integrate; default value : 100
parallel	a boolean to use parallelization, default FALSE

Value

a dataframe of Y real and noised values

Author(s)

Francois Bassac

Examples

```
df = generate_X_df(nind=100, curve_type = 'cat')
beta_real_func<-function(t, end_time=100, drop = NULL){
return(beta_real = sin(t*2*pi/end_time + pi) * exp (1.5*t/end_time))
}
```

```
beta_0_real=5.4321
Y_df = generate_Y_df(df, curve_type = 'cat',
beta_real_func, beta_0_real, NotS_ratio=0.2)
```

`generate_Y_df_CFD` *generate_Y_df_CFD*

Description

This function generates Y_df bases on df, beta_func with the following link Y = beta_0 + int(X(t)*beta(t))dt It generates also the noised values of Y. Here the NotS_ratio is the Noise over total Signal ratio meaning that a value of 0.2 means that the noise represents 20% of the TOTAL variance.

Usage

```
generate_Y_df_CFD(
  df,
  beta_real_func_or_list,
  beta_0_real = 5.4321,
  NotS_ratio = 0.2,
  id_col = "id",
  time_col = "time",
  int_mode = 1,
  nb_pt = 10,
  subdivisions = 100,
  seed = 123,
  parallel = FALSE
)
```

Arguments

`df`	the X(t) dataframe to evaluate Y on
`beta_real_func_or_list`	a function beta(t), or a list of function used for the Y evaluation
`beta_0_real`	the intercept, default 5.4321
`NotS_ratio`	the Noise over total Signal ratio, default 0.2
`id_col`	a character of the id column, default 'id'
`time_col`	a character of the time column, default 'time'
`int_mode`	integration mode, 1 for integrate, 2 for pracma::trapz
`nb_pt`	number of points for the integration, default value : 10
`subdivisions`	default parameter of R function integrate; default value : 100
`seed`	a integer, random seed
`parallel`	a boolean to enable parallel processing, default FALSE

Value

a dataframe of Y real and noised values

Author(s)

Francois Bassac

Examples

```
df = generate_X_df(nind=100, curve_type='cat')
beta_real_func<-function(t, end_time=100, drop = NULL){
return(beta_real = sin(t*2*pi/end_time + pi) * exp (1.5*t/end_time))
}
beta_0_real=5.4321
Y_df = generate_Y_df_CFD(df, beta_real_func, beta_0_real)
```

---

generate_Y_df_SFD *generate_Y_df_SFD*

Description

This function generates Y_df bases on df, beta_func with the following link Y = beta_0 + int(X(t)*beta(t))dt It generates also the noised values of Y. Here the NotS_ratio is the Noise over total Signal ratio meaning that a value of 0.2 means that the noise represents 20% of the TOTAL variance.

Usage

```
generate_Y_df_SFD(
  df,
  beta_real_func,
  beta_0_real = 5.4321,
  NotS_ratio = 0.2,
  id_col = "id",
  time_col = "time",
  seed = 123
)
```

Arguments

df	the X(t) dataframe to evaluate Y on
beta_real_func	a function beta(t), function used for the Y evaluation
beta_0_real	the intercept, default 5.4321
NotS_ratio	the Noise over total Signal ratio, default 0.2
id_col	a character of the id column, default 'id'
time_col	a character of the time column, default 'time'
seed	a integer, random seed, 123

Value

a dataframe

Author(s)

Francois Bassac

`gram_schmidt_orthonormalize` *gram_schmidt_orthonormalize*

Description

Orthonormalize a basis functions with Gram-Schmidt algorithm.

Usage

```
gram_schmidt_orthonormalize(basis, output_type = "fdlist", tol = 1e-12)
```

Arguments

`basis`	A basis object from fda package or a list of fd functions.
`output_type`	A character to choose the output format. "fdlist" (default) or "funlist" for R functions.
`tol`	a float, tolerance parameter, default 1e-12

Value

A list of orthonormalized functions fd or func(t)

Author(s)

Francois Bassac

Examples

```
start = 0
end = 10
basis = create_bspline_basis(start, end, nbasis = 10, norder = 4)

basis_orth = gram_schmidt_orthonormalize(basis, "fdlist")
```

help_smoothPLS *help_smoothPLS*

Description

This function print some information on the package use.

Usage

```
help_smoothPLS()
```

Value

No return value, called for side effects (prints instructions to the console).

Author(s)

Francois Bassac

initial_state_determination *initial_state_determination*

Description

This functions determine randomly the initial state of a categorical functional data, uniform probability between states.

Usage

```
initial_state_determination(N_states)
```

Arguments

N_states	a int the number of considered states

Value

a value of the designated state

Author(s)

Francois Bassac

Examples

```
initial_state_determination(2)
initial_state_determination(7)
```

is_orthogonal *is_orthogonal*

Description

Check if a basis function is orthogonal

Usage

```
is_orthogonal(basis, tol = 1e-10)
```

Arguments

basis	A basis object from fda package or a list of fd functions.
tol	a float, tolerance parameter, default 1e-10)

Value

A boolean, TRUE if orthogonal, FALSE if not

Author(s)

Francois Bassac

is_orthonormal *is_orthonormal*

Description

Check if a basis function is orthonormal

Usage

```
is_orthonormal(basis, tol = 1e-10)
```

Arguments

basis	A basis object from fda package or a list of fd functions.
tol	a float, tolerance parameter, default 1e-10)

Value

A boolean, TRUE if orthonormal, FALSE if not

Author(s)

Francois Bassac

lambda_determination *lambda_determination*

Description

This function determines a number of lambda parameters for exponential laws.

Usage

```
lambda_determination(N_states, lambda_values = c(0.05, 0.25))
```

Arguments

N_states	a int the number of states
lambda_values	a vector of min and max values authorized for lambda

Value

a numeric vector with the lambda values

Author(s)

Francois Bassac

Examples

```
lambda_determination(3)
lambda_determination(7)
```

mae_values *mae_values*

Description

This function evaluates the MAE error based on the values.

Usage

```
mae_values(y, y_hat)
```

Arguments

y	a vector of real values
y_hat	a vector of predicted values

Value

a value

Author(s)

Francois Bassac

Examples

```
y = c(1, 2, 4, 6, 8, 10)
y_hat = c(2, 3, 5, 7, 9, 11)
mae_values(y, y_hat)
```

`multivariate_alpha_building` *multivariate_alpha_building*

Description

This function builds the alphas matrix by columns binding the alpha of each curve. It returns the alphas matrix and a vector containing all the curves names.

Usage

```
multivariate_alpha_building(
  df_list,
  basis_list,
  curve_type_list,
  regul_time_list,
  id_col_list,
  time_col_list,
  print_steps = FALSE
)
```

Arguments

`df_list`	a list of data frame (id, time, state_or_value)
`basis_list`	a list of basis
`curve_type_list`	a list of curves
`regul_time_list`	a list of regul_time
`id_col_list`	a list if the characters for the id column
`time_col_list`	a list of the characters for the time column
`print_steps`	a boolean to print the current step, default FALSE

Value

a list of the alphas matrix and the curve_names vector and the new_basis_list

Author(s)

Francois Bassac

`multivariate_assemble_basis_metric`
multivariate_assemble_basis_metric

Description

This function assemble the metrics of all the basis of the basis list for multivariate use case. It take care on the categorical case by multiplying the basis per state.

Usage

```
multivariate_assemble_basis_metric(
  df_list,
  basis_list,
  curve_list,
  id_col_list,
  time_col_list
)
```

Arguments

`df_list`	a list of dataframe (id, time, value_or_state)
`basis_list`	a list of basis fd object
`curve_list`	a list of the curves type 'cat' or 'num'
`id_col_list`	a list of the character of the id columns
`time_col_list`	a list of the character of the time columns

Value

a matrix of the metric to consider

Author(s)

Francois Bassac

naivePLS *naivePLS*

Description

This function performs the naive PLS method for Categorical functional data, Scalar functional data and multivariate data.

Usage

```
naivePLS(
  df_list,
  Y,
  regul_time_obj = NULL,
  curve_type_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time",
  print_steps = FALSE,
  plot_rmsep = TRUE,
  print_nbComp = TRUE,
  plot_reg_curves = FALSE,
  validation = "LOO",
  jackknife = TRUE
)
```

Arguments

df_list	a list of dataframe (id, time, value_or_state)
Y	a numeric vector for the scalar response
regul_time_obj	a list of time regularization values
curve_type_obj	a list of the curve types 'cat' or 'num'
id_col_obj	a list of character of the names of the id columns
time_col_obj	a list of character of the names of the time columns
print_steps	a boolean to print the different steps, default FALSE
plot_rmsep	a boolean to plot the RMSEP, default TRUE
print_nbComp	a boolean to print the optimal number or components, default TRUE
plot_reg_curves	a boolean to plot the regression curves, default FALSE
validation	a character, pls::plsr input, default 'LOO'
jackknife	a boolean, pls::plsr input, default TRUE

Value

a list of ("plsr_model", "nbCP_opti", "curves_names", "opti_reg_coef", "reg_obj")

Author(s)

Francois Bassac

`naivePLS_formatting` *naivePLS_formatting*

Description

This function format the list of given dataframe into a time regularized matrix usable for the naive-PLS function or prediction.

Usage

```
naivePLS_formatting(
  df_list,
  regul_time_obj = NULL,
  curve_type_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time"
)
```

Arguments

`df_list`	a list of dataframe of 3 columns (id, time, state_or_value)
`regul_time_obj`	a list of vector of time regularization values
`curve_type_obj`	a list of character of the type of curves
`id_col_obj`	a list of character for the id column names
`time_col_obj`	a list of character for the time column names

Value

a list

Author(s)

Francois Bassacs

naivePLS_predict *naivePLS_predict*

Description

This function use the df_pred to make a prediction for a naivePLS object.

Usage

```
naivePLS_predict(
  naive_pls_obj,
  df_predict_list,
  regul_time_obj = NULL,
  curve_type_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time"
)
```

Arguments

naive_pls_obj	a list of a naivePLS object
df_predict_list	a list of dataframe (id, time, state_or_value)
regul_time_obj	a list of time regularization values
curve_type_obj	a list of curves types 'cat' or 'num'
id_col_obj	a list of id column names
time_col_obj	a list of time column names

Value

a numeric vector

Author(s)

Francois Bassac

number_of_test_id *number_of_test_id*

Description

This function evaluates the number of individuals for a test set.

Usage

```
number_of_test_id(TTRatio = 0.2, nind = 100)
```

Arguments

TTRatio	Train Test ratio, default 0.2
nind	number of individuals, default 100

Value

int, the number of id for test set

Author(s)

Francois Bassac

Examples

```
number_of_test_id(TTRatio = 0.2, nind=100)
```

obj_list_creation *obj_list_creation*

Description

This functions creates a list of basis containing the value of the number of states or curves sharing the same basis.

Usage

```
obj_list_creation(N_rep, obj)
```

Arguments

N_rep	a value of the number of states sharing the same basis
obj	a object

Value

a list of N_rep appended obj

Author(s)

Francois Bassac

Examples

```
N_rep = 4
start = 1
end = 51
nbasis = 13
norder = 3

basis = create_bspline_basis(start, end, nbasis, norder)
basis_list = obj_list_creation(N_rep, basis)
obj_list_creation(N_rep, 0:100)
```

---

orthonormalize_basis_list

orthonormalize_basis_list

Description

This function orthonormalized a basis from a list if needed.

Usage

```
orthonormalize_basis_list(basis_list, orth_list, tol = 1e-09)
```

Arguments

basis_list	a list of basis to orthonormalized
orth_list	a list of boolean per basis if orthonormalization is needed
tol	a float, tolerance parameter.

Value

a list of list of fd object : the orthonormalized functions.

Author(s)

Francois Bassac

plot_CFD_individuals *plot_CFD_individuals*

Description

This function only plot some individuals. It plots the first plot_individuals of df_to_plot. Works both for single state and multistates data (numerical states 1, 2, 3, etc)

Usage

```
plot_CFD_individuals(
  df_to_plot,
  n_ind_to_plot = 5,
  id_col = "id",
  time_col = "time",
  by_cfda = FALSE
)
```

Arguments

df_to_plot	dataframe whose individuals will be plotted
n_ind_to_plot	number of the first individuals to plot, default 5
id_col	col_name of df_to_plot for the id, not character, default id'
time_col	col_name of df_to_plot for the time, not character, default 'time'
by_cfda	a boolean to use cfda package function plotData, default FALSE

Value

a ggplot

Author(s)

Francois Bassac

Examples

```
df = generate_X_df()
plot_CFD_individuals(df, 5)
plot_CFD_individuals(df, 5, by_cfda = TRUE)
```

plot_fd_list *plot_fd_list*

Description

This function plots on the same figure the fd curves from the fd_list by evaluating them on the given regul_time

Usage

```
plot_fd_list(fd_list, curves_names, regul_time)
```

Arguments

fd_list	a list of fd objects
curves_names	a list of the curves names
regul_time	a numeric vector

Value

a plot

Author(s)

Francois Bassac

plot_model_metrics_base *plot_model_metrics_base*

Description

This function plots some histograms for train_results and test_results

Usage

```
plot_model_metrics_base(
  train_results,
  test_results,
  models_to_plot = c("FPLS", "SmoothPLS", "NaivePLS"),
  n_digits = 3
)
```

Arguments

train_results	a dataframe of train results
test_results	a dataframe of test results
models_to_plot	a list of characters of the models to plot, default c("FPLS", "SmoothPLS", "NaivePLS")
n_digits	a integer for the number of significant numbers to print, default 3

Value

a plot

Author(s)

Francois Bassac

---

plot_real_and_smoothed_data_ind
plot_real_and_smoothed_data_ind

Description

plot_real_and_smoothed_data_ind

Usage

```
plot_real_and_smoothed_data_ind(
  df_wide,
  df_fd,
  time_seq = 0:100,
  id = 1,
  col_list = c("blue", "red", "green", "yellow")
)
```

Arguments

df_wide	a wide dataframe, output of convert_to_wide_format()
df_fd	a list of functional data from df_wide
time_seq	a vector to plot on, default 0:100
id	a value of the id to plot
col_list	a list of color to separate the states

Value

No return value, called for side effects (generates a plot).

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas,  transition_df)
df_processed = cat_data_to_indicator(df)

df_regul = list()
df_wide = list()
for(name in names(df_processed)){
print(paste0(name, " regularisation"))
df_regul[[name]] = regularize_time_series(df_processed[[name]],
time_seq =  c(0:100), curve_type = 'cat', id_col='id', time_col='time')

df_wide[[name]] = convert_to_wide_format(df_regul[[name]], id_col='id',
time_col='time')
}

basis = create_bspline_basis(0, 100, 10, 4)

df_fd = list()
for(name in names(df_wide)){
print(paste0(name, " fd transformation"))
df_fd[[name]] = fda::Data2fd(argvals = c(0:100),
y = t(df_wide[[name]][, -c(1)]), basis)
}

plot_real_and_smoothed_data_ind(df_wide, df_fd, c(0:100), id=1)
```

`press_model` *press_model This function evaluates the PRESS error 'LOO' of a lm or glm model.*

Description

press_model This function evaluates the PRESS error 'LOO' of a lm or glm model.

Usage

```
press_model(model)
```

Arguments

model	a model from lm or glm

Value

a value

Author(s)

Francois Bassac

---

press_values *press_values*

Description

This function evaluates the press error using values y and y_hat

Usage

```
press_values(y, y_hat)
```

Arguments

y	a vector of real values
y_hat	a vector of predicted values

Value

a value

Author(s)

Francois Bassac

Examples

```
y = c(1, 2, 4, 6, 8, 10)
y_hat = c(2, 3, 5, 7, 9, 11)
press_values(y, y_hat)
```

p_w_building *p_w_building*

Description

This functions evaluates the p_i(t) (X_i regression functions) and w_i(t) such as $t_i = \int_0^T w_i(t) X_{i-1}(t) dt$. The evaluation is done for all the components.

Usage

```
p_w_building(
  coefficient,
  N_curves_processed,
  new_basis_list,
  new_orth_basis_list,
  curves_names_list
)
```

Arguments

coefficient	a table of the coefficient to use
N_curves_processed	a integer, the real number of curves (1 sfd = 1 curve, 1 cfd = 1curve per state)
new_basis_list	a list of processed basis list (1 basis per curve_processed)
new_orth_basis_list	a list of orthonormalized basis list (1 basis per curve_processed)
curves_names_list	a list of curve name (1 name per curve_processed)

Value

a list of fd objects

Author(s)

Francois Bassac

regularize_time_series *regularize_time_series*

Description

This function regularize the data on a new time interval time_seq given the curve_type. For curve_type = 'cat' the output state stay the same between 2 times. For curve_type = 'num' the intermediate values are interpolated linearly.

Usage

```
regularize_time_series(
  df,
  time_seq = 0:100,
  curve_type = NULL,
  id_col = "id",
  time_col = "time"
)
```

Arguments

df	dataframe with one or more different ids
time_seq	New time sequence where we want to regularize
curve_type	A string giving the type of the curve, 'cat' for a categorical functional data, 'num' for a scalar functional data, default : NULL
id_col	col_name of df for the id
time_col	col_name of df for the time

Value

a dataframe with the regularized data on time_seq

Author(s)

Francois Bassac

Examples

```
id_df = data.frame(id=rep(1,5), time=seq(0, 40, 10), state=c(0, 1, 1, 0, 1))
regularize_time_series(id_df, time_seq = seq(0, 40, 2), curve_type = 'cat')
regularize_time_series(id_df, time_seq = seq(0, 40, 2), curve_type = 'num')
```

`regularize_time_series_CFD` *regularize_time_series_CFD*

Description

This function regularize the data CFD on a new time interval time_seq

Usage

```
regularize_time_series_CFD(
  df,
  time_seq = 0:100,
  id_col = "id",
  time_col = "time"
)
```

Arguments

`df`	dataframe with one or more different ids
`time_seq`	New time sequence where we want to regularize
`id_col`	col_name of df for the id
`time_col`	col_name of df for the time

Value

a dataframe with the regularized data on time_seq

Author(s)

Francois Bassac

`regularize_time_series_SFD` *regularize_time_series_SFD*

Description

This function regularizes a Scalar Functional Data SFD on the time_seq input. This function uses linear interpolation.

Usage

```
regularize_time_series_SFD(
  df,
  time_seq = 0:100,
  id_col = "id",
  time_col = "time"
)
```

Arguments

df	dataframe with one or more different ids
time_seq	New time sequence where we want to regularize
id_col	col_name of df for the id
time_col	col_name of df for the time

Value

a regularized dataframe

Author(s)

Francois Bassac

---

reg_curve_funcPLS_evaluation *reg_curve_funcPLS_evaluation*

Description

This function evaluate the beta regression functions of the multivariate functional PLS.

Usage

```
reg_curve_funcPLS_evaluation(
  mfpls_mfd,
  nb_comp_pls_opt,
  root_metric,
  new_basis_list,
  curve_names,
  print_steps = FALSE
)
```

Arguments

mfpls_mfd	a MFPLS model
nb_comp_pls_opt	the optimal number of components
root_metric	the root metric
new_basis_list	a list of basis fd objects
curve_names	a list of curves to keep
print_steps	a boolean to cat or not the steps

Value

a list of the betas per state

Author(s)

Francois Bassac

---

remove_duplicate_states *remove_duplicate_states*

Description

This function removes the duplicated states and keep the last line. this function works both with (0,1) or categorical states on the state_col!

Usage

```
remove_duplicate_states(data, id_col = "id", time_col = "time")
```

Arguments

data	a single- or multi-state dataframe.
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'

Value

a dataframe with no duplicated states.

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas, transition_df)
remove_duplicate_states(df)
```

`r_squared_values` *r_squared_values*

Description

This function evaluates the R_2 using values y and y_hat

Usage

```
r_squared_values(y, y_hat)
```

Arguments

y	a vector of real values
y_hat	a vector of predicted values

Value

a value

Author(s)

Francois Bassac

Examples

```
y = c(1, 2, 4, 6, 8, 10)
y_hat = c(2, 3, 5, 7, 9, 11)
r_squared_values(y, y_hat)
```

select_from_fd_list *select_from_fd_list*

Description

This function return a list of functions of fd_list without the ones specified in the numeric vector toDrop.

Usage

```
select_from_fd_list(fd_list, toDrop = NULL)
```

Arguments

fd_list	a list of fd object
toDrop	a numeric vector

Value

a list of fd objects

Author(s)

Francois Bassac

smoothPLS *smoothPLS_multi*

Description

This function performs the smooth PLS algorithm for both the univariate case for a Scalar Functional Data or a Categorical Functional Data and the multivariate case for a mix of functional data of different nature (SFD or CFD). For some input, if the same value is needed for all the different curves, no need to make a list with the value per curve.

Usage

```
smoothPLS(
  df_list,
  Y,
  basis_obj,
  regul_time_obj = NULL,
  curve_type_obj,
  orth_obj = TRUE,
  id_col_obj = "id",
```

```
  time_col_obj = "time",
  int_mode = 1,
  print_steps = FALSE,
  plot_rmsep = TRUE,
  print_nbComp = TRUE,
  plot_reg_curves = FALSE,
  jackknife = TRUE,
  validation = "LOO",
  parallel = TRUE
)
```

Arguments

df_list	a list of dataframe (id, time, value_or_state)
Y	a vector of the scalar response
basis_obj	a basis fd object or a list of basis fd object
regul_time_obj	a vector for time regularization values or a list
curve_type_obj	a list or vector of the different curves types, 'cat' or 'num.
orth_obj	a list or a vector of booleans if the orthonormalization is needed
id_col_obj	a list or a vector of the id column name
time_col_obj	a list or a vector of time column name
int_mode	a integer of the integration method : 1 for integrate, 2 for pracma::trapz
print_steps	a boolean to print the algorithm steps
plot_rmsep	a boolean to plot the pls model RMSEP
print_nbComp	a boolean to print the optimal number of components
plot_reg_curves	a boolean to plot the regressions curves
jackknife	a boolean for the jackknife input of pls() function, default TRUE
validation	a character for the validation input of pls() function, default 'LOO'
parallel	a boolean to use parallelization, default TRUE

Value

a list of the plsr_model and the regression curves (and intercept).

Author(s)

Francois Bassac

`smoothPLS_CFD_predict` *smoothPLS_CFD_predict (Updated v0.1.4)*

Description

Predicts the response variable for Categorical Functional Data (CFD) by integrating the regression coefficient function over active state intervals.

Usage

```
smoothPLS_CFD_predict(
  df_predict,
  delta_spls,
  id_col = "id",
  time_col = "time",
  subdivisions = 100,
  parallel = TRUE,
  ...
)
```

Arguments

`df_predict`	Dataframe containing columns for id, time, and state.
`delta_spls`	A list containing the scalar intercept and the functional regression coefficient (fd object).
`id_col`	Character, name of the id column, default 'id'.
`time_col`	Character, name of the time column, default 'time'.
`subdivisions`	integer, maximum number of sub-intervals for integration, default 100
`parallel`	a boolean to enable parallel processing, default TRUE.
`...`	Additional parameters passed to evaluate_id_func_integral (e.g., rel_tol, subdivisions).

Value

A numeric vector of predicted values for each individual.

Author(s)

Francois Bassac

`smoothPLS_CFD_predict_para_v1` *smoothPLS_CFD_predict_para_v1 (Updated v0.1.2)*

Description

Predicts the response variable for Categorical Functional Data (CFD) by integrating the regression coefficient function over active state intervals.

Usage

```
smoothPLS_CFD_predict_para_v1(
  df_predict,
  delta_spls,
  id_col = "id",
  time_col = "time",
  subdivisions = 100,
  parallel = TRUE,
  ...
)
```

Arguments

`df_predict`	Dataframe containing columns for id, time, and state.
`delta_spls`	A list containing the scalar intercept and the functional regression coefficient (fd object).
`id_col`	Character, name of the id column, default 'id'.
`time_col`	Character, name of the time column, default 'time'.
`subdivisions`	integer, maximum number of sub-intervals for integration, default 100
`parallel`	a boolean to enable parallel processing, default TRUE.
`...`	Additional parameters passed to evaluate_id_func_integral (e.g., rel_tol, subdivisions).

Value

A numeric vector of predicted values for each individual.

Author(s)

Francois Bassac

smoothPLS_predict *smoothPLS_multi_predict*

Description

This function use the list of regression functions to make a prediction $\hat{Y} = \delta_0 + \int_0^T \delta(t)X(t)dt$

Usage

```
smoothPLS_predict(
  df_predict_list,
  delta_list,
  curve_type_obj = NULL,
  id_col_obj = "id",
  time_col_obj = "time",
  regul_time_obj = NULL,
  int_mode = 1,
  nb_pt = 10,
  subdivisions = 100,
  parallel = TRUE
)
```

Arguments

df_predict_list	a list of dataframe (id, time, value_or_state)
delta_list	a list of regression object (intercept, delta_1_fd, delta_2_fd, etc)
curve_type_obj	a list of characters of the curve types 'cat' or 'num'
id_col_obj	a list of character of the name of the id column, default 'id'
time_col_obj	a list of character of the name of the time column, default 'time'
regul_time_obj	a list of the time regularization values
int_mode	a integer for the integration mode, 1 for integrate, 2 for pracma::trapz
nb_pt	a integer, number of intermediate points for pracma::trapz, default 10
subdivisions	a integer, number of subdivision in integrate function, default 100
parallel	a boolean to use parallelization, default TRUE

Value

a numeric vector of the prediction

Author(s)

Francois Bassac

smoothPLS_predict_uni *smoothPLS_predict_uni*

Description

This function make a prediction base on a dataframe and a list made of the intercept and the regression curve. The input curve_type in needed to select the good way of evaluate the integrals $\int_0^T delta(t)X(t)dt$.

Usage

```
smoothPLS_predict_uni(
  df_predict,
  delta_list,
  curve_type = NULL,
  int_mode = 1,
  id_col = "id",
  time_col = "time",
  nb_pt = 10,
  subdivisions = 100,
 regul_time = seq(delta_list[[2]]$basis$rangeval[1], delta_list[[2]]$rangeval[2], 1),
  parallel = TRUE
)
```

Arguments

df_predict	a dataframe ('id', 'time', 'state or value') to predict from
delta_list	a list of delta_spls : list(intercept, delta_fd)
curve_type	a character, 'cat' for Categorical FD, 'num' for Scalar FD
int_mode	a value of the integration mode, default 1
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'
nb_pt	number of points for the integration, default value : 10
subdivisions	default parameter of R function integrate; default value : 100
regul_time	a vector of time regularization values default delta_fd basis rangeval per 1, NEEDED for curve_type = 'num'!
parallel	a boolean to use parallelization, default TRUE

Value

a vector of predicted values $\hat{Y} = \delta_0 + \int_0^T X(t)\delta(t)dt$

Author(s)

Francois Bassac

`smoothPLS_SFD_predict` *smoothPLS_SFD_predict*

Description

Predicts the response Y for Scalar Functional Data using the analytic L2 inner product. This implementation follows the theory from Chapter 7.

Usage

```
smoothPLS_SFD_predict(
  df_predict,
  delta_spls,
  basis_obj = NULL,
  id_col = "id",
  time_col = "time",
  parallel = TRUE,
  ...
)
```

Arguments

`df_predict`	Dataframe with columns (id, time, value).
`delta_spls`	List containing (intercept, delta_fd_object).
`basis_obj`	Optional basis for signal reconstruction. If NULL, uses the basis from delta_fd
`id_col`	Character, name of id column.
`time_col`	Character, name of time column.
`parallel`	a boolean to enable parallel processing, default TRUE.
`...`	Additional arguments for Data2fd or inprod.

Value

A numeric vector of predicted values

Author(s)

Francois Bassac

smoothPLS_SFD_predict_para_v1 *smoothPLS_SFD_predict_para_v1*

Description

Predicts the response Y for Scalar Functional Data using the analytic L2 inner product. This implementation follows the theory from Chapter 7.

Usage

```
smoothPLS_SFD_predict_para_v1(
  df_predict,
  delta_spls,
  basis_obj = NULL,
  id_col = "id",
  time_col = "time",
  parallel = TRUE,
  ...
)
```

Arguments

df_predict	Dataframe with columns (id, time, value).
delta_spls	List containing (intercept, delta_fd_object).
basis_obj	Optional basis for signal reconstruction. If NULL, uses the basis from delta_fd
id_col	Character, name of id column.
time_col	Character, name of time column.
parallel	a boolean to enable parallel processing, default TRUE.
...	Additional arguments for Data2fd or inprod.

Value

A numeric vector of predicted values

Author(s)

Francois Bassac

split_in_state_df *split_in_state_df*

Description

This function transform a categorical functional data with its indicator functions into a dedicated list of all the state (one per different state) This function will also work with character states.

Usage

```
split_in_state_df(data, id_col = "id", time_col = "time")
```

Arguments

data	a dataframe containing the indicator functions, output of state_indicator()
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'

Value

a list containing the dataframe of the indicator function of each state.

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas, transition_df)

si_df = state_indicator(df, id_col='id',
time_col='time')

split_df = split_in_state_df(si_df, id_col='id', time_col='time')
```

state_indicator *state_indicator*

Description

This function takes functional categorical curve as input and transform it into as many indicator curves as the number of state input return DATAFRAME Works even on dataframe without time condition respected (same start and end) This function sort the states by ascending order (if numeric) and put the name 'state_X' as the column of the output concerning the 'X' state. This function will also work with character states. Now for the different lists, the ith element of a list concern the ith states ordered.

Usage

```
state_indicator(data, id_col = "id", time_col = "time")
```

Arguments

data	a multistates dataframe ('id', 'time', 'states')
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'

Value

a dataframe with columns ('id', 'time', list of states_XX)

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas, transition_df)

si_df = state_indicator(df, id_col='id',
time_col='time')
```

state_indicator_old *state_indicator_old*

Description

This function takes functional categorical curve as input and transform it into as many indicator curves as the number of state input return DATAFRAME Works even on dataframe without time condition respected (same start and end) This function sort the states by ascending order (if numeric) and put the name 'state_X' as the column of the output concerning the 'X' state. This function will also work with character states. Now for the different lists, the ith element of a list concern the ith states ordered.

Usage

```
state_indicator_old(data, id_col = "id", time_col = "time")
```

Arguments

data	a multistates dataframe ('id', 'time', 'states')
id_col	a character for the id column, default 'id'
time_col	a character for the time column, default 'time'

Value

a dataframe with columns ('id', 'time', list of states_XX)

Author(s)

Francois Bassac

Examples

```
N_states = 3
lambdas = lambda_determination(N_states)
transition_df = transfer_probabilities(N_states)

df = generate_X_df_multistates(nind = 100, N_states, start=0, end=100,
lambdas, transition_df)

si_df = state_indicator_old(df, id_col='id',
time_col='time')
```

test_basis_properties *test_basis_properties*

Description

General function to test basis functions

Usage

```
test_basis_properties(basis, name = "Base", tol = 1e-10)
```

Arguments

basis	Basis to test, basis object or list of fd functions
name	Character, name of the basis
tol	Float, precision, default 1e-10

Value

a boolean

Author(s)

Francois Bassac

Examples

```
start = 0
end = 10
basis = create_bspline_basis(start, end, nbasis=10, norder=4)
test_basis_properties(basis, "cubic splines", tol = 1e-3)

basis_f = fda::create.fourier.basis(rangeval=c(start, end), nbasis=5)
test_basis_properties(basis_f, "Fourier", tol = 1e-3)
```

transfer_probabilities *transfer_probabilities*

Description

This function gives transfer probabilities between states. row -> columns.

Usage

```
transfer_probabilities(N_states)
```

Arguments

N_states	a int the number a states considered.

Value

a dataframe containing the transition probabilities

Author(s)

Francois Bassac

Examples

```
transfer_probabilities(3)
transfer_probabilities(5)
```

---

transition_matrix *transition_matrix*

Description

Build transition matrix between to basis.

Usage

```
transition_matrix(basis1, basis2)
```

Arguments

basis1	First basis, basis obj or list of fd functions
basis2	Second basis, basis obj or list of fd functions

Value

Transition matrix P such as basis1 = P * basis2

Author(s)

Francois Bassac

`univariate_alpha_building` *univariate_alpha_building*

Description

This function build the alphas matrix for a dataframe of a curve_type curve on a basis on a regul_time vector.

Usage

```
univariate_alpha_building(
  df,
  basis,
  curve_type = NULL,
  regul_time,
  id_col = "id",
  time_col = "time"
)
```

Arguments

`df`	a dataframe (id, time, value_or_state)
`basis`	a basis fd object
`curve_type`	a character, 'cat' or 'num'
`regul_time`	a numeric vector for time regularization
`id_col`	a character for the id column name
`time_col`	a character for the time column name

Value

a matrix

Author(s)

Francois Bassac

Index

assemble_basis_metric, 4
assert_funcPLS_inputs, 5
assert_multivariate_naivePLS_inputs, 6
assert_multivariate_smoothPLS_inputs, 7

beta_1_real_func, 8
beta_2_real_func, 8
beta_3_real_func, 9
beta_4_real_func, 10
beta_5_real_func, 11
beta_6_real_func, 11
beta_7_real_func, 12
beta_list_generation, 13
block_diag, 14
build_df_per_state, 14
build_new_data_list, 15
build_reg_curve_spls, 16
build_spls_functions, 17
build_u_ki_list, 18

cat_data_to_indicator, 18
convert_to_wide_format, 19
create_bspline_basis, 20

determine_curve_name, 21
determine_next_state, 21

eval_max_min_y, 34
evaluate_curves_distances, 22
evaluate_gamma_ij, 23
evaluate_id_func_integral, 23
evaluate_id_func_integral_deprecated, 24
evaluate_lambda, 25
evaluate_lambda_CFD, 26
evaluate_lambda_CFD_para_v1, 27
evaluate_lambda_SFD, 28
evaluate_lambda_SFD_para_v1, 29
evaluate_metric, 30
evaluate_reg_curve_SPLS_uni, 31
evaluate_results, 32
evaluate_V_i_function, 33
evaluate_variance_explained, 32

from_basis_to_fdlist, 34
from_fd_to_func, 35
funcPLS, 36
funcPLS_predict, 37

generate_probabilities, 38
generate_X_df, 39
generate_X_df_CFD, 40
generate_X_df_multistates, 41
generate_X_df_SFD, 42
generate_X_df_SFD_data, 42
generate_X_df_test, 43
generate_Y_df, 44
generate_Y_df_CFD, 46
generate_Y_df_SFD, 47
gram_schmidt_orthonormalize, 48

help_smoothPLS, 49

initial_state_determination, 49
is_orthogonal, 50
is_orthonormal, 50

lambda_determination, 51

mae_values, 51
multivariate_alpha_building, 52
multivariate_assemble_basis_metric, 53

naivePLS, 54
naivePLS_formatting, 55
naivePLS_predict, 56
number_of_test_id, 57

obj_list_creation, 57
orthonormalize_basis_list, 58

p_w_building, 64
plot_CFD_individuals, 59
plot_fd_list, 60
plot_model_metrics_base, 60
plot_real_and_smoothed_data_ind, 61
press_model, 62
press_values, 63

r_squared_values, 69
reg_curve_funcPLS_evaluation, 67
regularize_time_series, 65
regularize_time_series_CFD, 66
regularize_time_series_SFD, 66
remove_duplicate_states, 68

select_from_fd_list, 70
smoothPLS, 70
smoothPLS_CFD_predict, 72
smoothPLS_CFD_predict_para_v1, 73
smoothPLS_predict, 74
smoothPLS_predict_uni, 75
smoothPLS_SFD_predict, 76
smoothPLS_SFD_predict_para_v1, 77
split_in_state_df, 78
state_indicator, 79
state_indicator_old, 80

test_basis_properties, 81
transfer_probabilities, 81
transition_matrix, 82

univariate_alpha_building, 83