Package ‘funcharts’

October 13, 2022

Type Package

Title Functional Control Charts

Version 1.2.0


Depends fda, dplyr, ggplot2, R (>= 3.6.0)
Imports rlang, parallel, tidyr, patchwork, RSpectra, matrixStats
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.2.0
Suggests knitr, rmarkdown, testthat
VignetteBuilder knitr

URL https://github.com/unina-sfere/funcharts

BugReports https://github.com/unina-sfere/funcharts/issues

NeedsCompilation no

Author Christian Capezza [cre, aut], Fabio Centofanti [aut], Antonio Lepore [aut], Alessandra Menafoglio [aut], Biagio Palumbo [aut], Simone Vantini [aut]

Maintainer Christian Capezza <christian.capezza@unina.it>

Repository CRAN

Date/Publication 2022-06-15 08:20:08 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>topic</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>air</td>
<td>3</td>
</tr>
<tr>
<td>cbind_mfd</td>
<td>4</td>
</tr>
<tr>
<td>control_charts_pca</td>
<td>4</td>
</tr>
<tr>
<td>control_charts_pca_mfd_real_time</td>
<td>7</td>
</tr>
<tr>
<td>control_charts_sof_pc</td>
<td>9</td>
</tr>
<tr>
<td>control_charts_sof_pc_real_time</td>
<td>11</td>
</tr>
<tr>
<td>cont_plot</td>
<td>13</td>
</tr>
<tr>
<td>data_sim_mfd</td>
<td>14</td>
</tr>
<tr>
<td>fof_pc</td>
<td>15</td>
</tr>
<tr>
<td>fof_pc_real_time</td>
<td>17</td>
</tr>
<tr>
<td>funcharts</td>
<td>19</td>
</tr>
<tr>
<td>geom_mfd</td>
<td>19</td>
</tr>
<tr>
<td>get_mfd_array</td>
<td>20</td>
</tr>
<tr>
<td>get_mfd_array_real_time</td>
<td>21</td>
</tr>
<tr>
<td>get_mfd_df</td>
<td>23</td>
</tr>
<tr>
<td>get_mfd_df_real_time</td>
<td>25</td>
</tr>
<tr>
<td>get_mfd_fd</td>
<td>26</td>
</tr>
<tr>
<td>get_mfd_list</td>
<td>27</td>
</tr>
<tr>
<td>get_mfd_list_real_time</td>
<td>28</td>
</tr>
<tr>
<td>get_ooc</td>
<td>30</td>
</tr>
<tr>
<td>get_sof_pc_outliers</td>
<td>31</td>
</tr>
<tr>
<td>inprod_mfd</td>
<td>31</td>
</tr>
<tr>
<td>inprod_mfd_diag</td>
<td>32</td>
</tr>
<tr>
<td>is.mfd</td>
<td>32</td>
</tr>
<tr>
<td>mfd</td>
<td>33</td>
</tr>
<tr>
<td>norm.mfd</td>
<td>33</td>
</tr>
<tr>
<td>pca_mfd</td>
<td>35</td>
</tr>
<tr>
<td>pca_mfd_real_time</td>
<td>36</td>
</tr>
<tr>
<td>plot_bifd</td>
<td>37</td>
</tr>
<tr>
<td>plot_bootstrap_sof_pc</td>
<td>38</td>
</tr>
<tr>
<td>plot_control_charts</td>
<td>39</td>
</tr>
<tr>
<td>plot_control_charts_real_time</td>
<td>40</td>
</tr>
<tr>
<td>plot_mfd</td>
<td>41</td>
</tr>
<tr>
<td>plot_mon</td>
<td>42</td>
</tr>
<tr>
<td>plot_pca_mfd</td>
<td>43</td>
</tr>
<tr>
<td>predict_fof_pc</td>
<td>44</td>
</tr>
<tr>
<td>predict_sof_pc</td>
<td>45</td>
</tr>
<tr>
<td>rbind_mfd</td>
<td>46</td>
</tr>
<tr>
<td>regr_cc_fof</td>
<td>47</td>
</tr>
<tr>
<td>regr_cc_fof_real_time</td>
<td>49</td>
</tr>
<tr>
<td>regr_cc_sof</td>
<td>51</td>
</tr>
<tr>
<td>scale_mfd</td>
<td>52</td>
</tr>
<tr>
<td>simulate_mfd</td>
<td>53</td>
</tr>
<tr>
<td>sim_funcharts</td>
<td>55</td>
</tr>
<tr>
<td>sof_pc</td>
<td>56</td>
</tr>
<tr>
<td>sof_pc_real_time</td>
<td>58</td>
</tr>
</tbody>
</table>
Description

This data set has been included from the R package FRegSigCom. The original .RData file is available at https://github.com/cran/FRegSigCom/blob/master/data/air.RData.

Data collected hourly in 355 days (days with missing values removed) in a significantly polluted area within an Italian city.

Usage

data("air")

Format

A list of 7 matrices with 355 rows and 24 columns:

- **NO2** Hourly observation of concentration level of NO2 in 355 days
- **CO** Hourly observation of concentration level of CO in 355 days
- **NMHC** Hourly observation of concentration level of NMHC in 355 days
- **NOx** Hourly observation of concentration level of NOx in 355 days
- **C6H6** Hourly observation of concentration level of C6H6 in 355 days
- **temperature** Hourly observation of concentration level of temperature in 355 days
- **humidity** Hourly observation of concentration level of humidity in 355 days

Source

https://archive.ics.uci.edu/ml/datasets/Air+quality

References


cbind_mfd  

Bind variables of two Multivariate Functional Data Objects

Description

Bind variables of two Multivariate Functional Data Objects

Usage

\[
cbind_mfd(mfdobj1, mfdobj2)
\]

Arguments

- \textit{mfdobj1}: An object of class \textit{mfd}, with the same number of replications of \textit{mfdobj2} and different variable names with respect to \textit{mfdobj2}.
- \textit{mfdobj2}: An object of class \textit{mfd}, with the same number of replications of \textit{mfdobj1}, and different variable names with respect to \textit{mfdobj1}.

Value

An object of class \textit{mfd}, whose replications are the same of \textit{mfdobj1} and \textit{mfdobj2} and whose functional variables are the union of the functional variables in \textit{mfdobj1} and \textit{mfdobj2}.

Examples

```r
library(funcharts)
mfdobj1 <- data_sim_mfd(nvar = 3)
mfdobj2 <- data_sim_mfd(nvar = 2)
dimnames(mfdobj2$coefs)[[3]] <- mfdobj2$fdnames[[3]] <- c("var10", "var11")

plot_mfd(mfdobj1)
plot_mfd(mfdobj2)
mfdobj_cbind <- cbind_mfd(mfdobj1, mfdobj2)
plot_mfd(mfdobj_cbind)
```

control_charts_pca  

\(T^2\) and \(SPE\) control charts for multivariate functional data

Description

This function builds a data frame needed to plot the Hotelling’s \(T^2\) and squared prediction error (SPE) control charts based on multivariate functional principal component analysis (MFPCA) performed on multivariate functional data, as Capezza et al. (2020) for the multivariate functional covariates. The training data have already been used to fit the model. An optional tuning data set can be provided to estimate the control chart limits. A phase II data set contains the observations to be monitored with the control charts.
control_charts_pca

Usage

control_charts_pca(
  pca,
  components = NULL,
  tuning_data = NULL,
  newdata,
  alpha = list(T2 = 0.025, spe = 0.025),
  limits = "standard",
  seed,
  nfold = 5,
  ncores = 1,
  tot_variance_explained = 0.9,
  single_min_variance_explained = 0
)

Arguments

pca An object of class pca_mfd obtained by doing MFPCA on the training set of multivariate functional data.

components A vector of integers with the components over which to project the multivariate functional data. If this is not NULL, the arguments 'single_min_variance_explained' and 'tot_variance_explained' are ignored. If NULL, components are selected such that the total fraction of variance explained by them is at least equal to the argument 'tot_variance_explained', where only components explaining individually a fraction of variance at least equal to the argument 'single_min_variance_explained' are considered to be retained. Default is NULL.

tuning_data An object of class mfd containing the tuning set of the multivariate functional data, used to estimate the T^2 and SPE control chart limits. If NULL, the training data, i.e. the data used to fit the MFPCA model, are also used as the tuning data set, i.e. tuning_data=pca$data. Default is NULL.

newdata An object of class mfd containing the phase II set of the multivariate functional data to be monitored.

alpha A named list with two elements, named T2 and spe, respectively, each containing the desired Type I error probability of the corresponding control chart. Note that at the moment you have to take into account manually the family-wise error rate and adjust the two values accordingly. See Capezza et al. (2020) and Centofanti et al. (2021) for additional details. Default value is list(T2 = 0.025, spe = 0.025).

limits A character value. If "standard", it estimates the control limits on the tuning data set. If "cv", the function calculates the control limits only on the training data using cross-validation using calculate_cv_limits. Default is "standard".

seed If limits="cv", since the split in the k groups is random, you can fix a seed to ensure reproducibility. Deprecated: use set.seed() before calling the function for reproducibility.

nfold If limits="cv", this gives the number of groups k used for k-fold cross-validation. If it is equal to the number of observations in the training data set,
then we have leave-one-out cross-validation. Otherwise, this argument is ignored.

**ncores**
If limits=="cv", if you want perform the analysis in the k groups in parallel, give the number of cores/threads. Otherwise, this argument is ignored.

**tot_variance_explained**
The minimum fraction of variance that has to be explained by the set of multivariate functional principal components retained into the MFPCA model fitted on the functional covariates. Default is 0.9.

**single_min_variance_explained**
The minimum fraction of variance that has to be explained by each multivariate functional principal component such that it is retained into the MFPCA model. Default is 0.

**Value**
A data.frame with as many rows as the number of multivariate functional observations in the phase II data set and the following columns:
* one id column identifying the multivariate functional observation in the phase II data set,
* one T2 column containing the Hotelling T^2 statistic calculated for all observations,
* one column per each functional variable, containing its contribution to the T^2 statistic,
* one spe column containing the SPE statistic calculated for all observations,
* one column per each functional variable, containing its contribution to the SPE statistic,
* T2_lim gives the upper control limit of the Hotelling’s T^2 control chart,
* one contribution_T2_*_lim column per each functional variable giving the limits of the contribution of that variable to the Hotelling’s T^2 statistic,
* spe_lim gives the upper control limit of the SPE control chart
* one contribution_spe_*_lim column per each functional variable giving the limits of the contribution of that variable to the SPE statistic.

**References**

**See Also**
regr_cc_fof

**Examples**
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:220, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates],
```r
n_basis = 15,
lambda = 1e-2)
y <- rowMeans(air$NO2)
y1 <- y[1:100]
y_tuning <- y[101:200]
y2 <- y[201:220]
mfdobj_x1 <- mfdobj_x[1:100]
mfdobj_x_tuning <- mfdobj_x[101:200]
mfdobj_x2 <- mfdobj_x[201:220]
pca <- pca_mfd(mfdobj_x1)
cclist <- control_charts_pca(pca = pca,
    tuning_data = mfdobj_x_tuning,
    newdata = mfdobj_x2)
plot_control_charts(cclist)
```

---

**control_charts_pca_mfd_real_time**

*Real-time $T^2$ and SPE control charts for multivariate functional data*

**Description**

This function produces a list of data frames, each of them is produced by `control_charts_pca` and is needed to plot control charts for monitoring multivariate functional covariates each evolving up to an intermediate domain point.

**Usage**

```r
control_charts_pca_mfd_real_time(
    pca_list,
    components_list = NULL,
    mfdobj_x_test,
    mfdobj_x_tuning = NULL,
    alpha = list(T2 = 0.0125, spe = 0.0125),
    limits = "standard",
    seed,
    nfold = NULL,
    tot_variance_explained = 0.9,
    single_min_variance_explained = 0,
    ncores = 1
)
```

**Arguments**

- `pca_list` A list of lists produced by `pca_mfd_real_time`, containing a list of multivariate functional principal component analysis models estimated on functional data each evolving up to an intermediate domain point.
components_list
A list of components given as input to pca_mfd for each intermediate domain point.

mfdobj_x_test
A list created using get_mfd_df_real_time or get_mfd_list_real_time, denoting a list of functional data objects in the phase II monitoring data set, each evolving up to an intermediate domain point, with observations of the multivariate functional data. The length of this list and pca_list must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point.

mfdobj_x_tuning
A list created using get_mfd_df_real_time or get_mfd_list_real_time, denoting a list of functional data objects in the tuning data set (used to estimate control chart limits), each evolving up to an intermediate domain point, with observations of the multivariate functional data. The length of this list and pca_list must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point. If NULL, the training data, i.e., the functional data in pca_list, are also used as the tuning data set. Default is NULL.

alpha
See control_charts_pca.

limits
See control_charts_pca.

seed
Deprecated: See control_charts_pca.

nfold
See control_charts_pca.

tot_variance_explained
See control_charts_pca.

single_min_variance_explained
See control_charts_pca.

ncores
If you want parallelization, give the number of cores/threads to be used when creating objects separately for different instants.

Value
A list of data.frames each produced by control_charts_pca, corresponding to a given instant.

See Also
pca_mfd_real_time, control_charts_pca

Examples
library(funcharts)
data("air")
air1 <- lapply(air, function(x) x[1:8, , drop = FALSE])
air2 <- lapply(air, function(x) x[9:10, , drop = FALSE])
mfdobj_x1_list <- get_mfd_list_real_time(air1[c("CO", "temperature")],
n_basis = 15,
lambda = 1e-2,
k_seq = c(0.5, 1))
mfdobj_x2_list <- get_mfd_list_real_time(air2[c("CO", "temperature")],

control_charts_sof_pc

n_basis = 15,
lambda = 1e-2,
k_seq = c(0.5, 1))

pca_list <- pca_mfd_real_time(mfdobj_x1_list)

cclist <- control_charts_pca_mfd_real_time(
  pca_list = pca_list,
  components_list = 1:3,
  mfdobj_x_test = mfdobj_x2_list)

plot_control_charts_real_time(cclist, 1)

control_charts_sof_pc  Control charts for monitoring a scalar quality characteristic adjusted for by the effect of multivariate functional covariates

Description

This function builds a data frame needed to plot control charts for monitoring a scalar quality characteristic adjusted for the effect of multivariate functional covariates based on scalar-on-function regression, as proposed in Capezza et al. (2020).

In particular, this function provides:
* the Hotelling’s $T^2$ control chart,
* the squared prediction error (SPE) control chart,
* the scalar regression control chart.

This function calls control_charts_pca for the control charts on the multivariate functional covariates and regr_cc_sof for the scalar regression control chart.

The training data have already been used to fit the model. An optional tuning data set can be provided that is used to estimate the control chart limits. A phase II data set contains the observations to be monitored with the control charts.

Usage

control_charts_sof_pc(
  mod,
  y_test,
  mfdobj_x_test,
  mfdobj_x_tuning = NULL,
  alpha = list(T2 = 0.0125, spe = 0.0125, y = 0.025),
  limits = "standard",
  seed,
  nfold = NULL,
  ncores = 1
)
Arguments

mod  A list obtained as output from sof_pc, i.e. a fitted scalar-on-function linear regression model.
y_test  A numeric vector containing the observations of the scalar response variable in the phase II data set.
mfdobj_x_test  An object of class mfd containing the phase II data set of the functional covariates observations.
mfdobj_x_tuning  An object of class mfd containing the tuning set of the multivariate functional data, used to estimate the T^2 and SPE control chart limits. If NULL, the training data, i.e. the data used to fit the MFPCA model, are also used as the tuning data set, i.e. tuning_data=pca$data. Default is NULL.
alpha  A named list with three elements, named T2, spe, and codey, respectively, each containing the desired Type I error probability of the corresponding control chart (T2 corresponds to the T^2 control chart, spe corresponds to the SPE control chart, y corresponds to the scalar regression control chart). Note that at the moment you have to take into account manually the family-wise error rate and adjust the two values accordingly. See Capezza et al. (2020) for additional details. Default value is list(T2 = 0.0125, spe = 0.0125, y = 0.025).
limits  A character value. If "standard", it estimates the control limits on the tuning data set. If "cv", the function calculates the control limits only on the training data using cross-validation using calculate_cv_limits. Default is "standard".
seed  If limits=="cv", since the split in the k groups is random, you can fix a seed to ensure reproducibility. Deprecated: use set.seed() before calling the function for reproducibility.
nfold  If limits=="cv", this gives the number of groups k used for k-fold cross-validation. If it is equal to the number of observations in the training data set, then we have leave-one-out cross-validation. Otherwise, this argument is ignored.
ncores  If limits=="cv", if you want perform the analysis in the k groups in parallel, give the number of cores/threads. Otherwise, this argument is ignored.

Value

A data.frame with as many rows as the number of multivariate functional observations in the phase II data set and the following columns:
* one id column identifying the multivariate functional observation in the phase II data set,
* one T2 column containing the Hotelling T^2 statistic calculated for all observations,
* one column per each functional variable, containing its contribution to the T^2 statistic,
* one spe column containing the SPE statistic calculated for all observations,
* one column per each functional variable, containing its contribution to the SPE statistic,
* T2_lim gives the upper control limit of the Hotelling’s T^2 control chart,
* one contribution_T2_*_lim column per each functional variable giving the limits of the contribution of that variable to the Hotelling’s T^2 statistic,
control_charts_sof_pc_real_time

Real-time scalar-on-function regression control charts

Description

This function produces a list of data frames, each of them is produced by control_charts_sof_pc and is needed to plot control charts for monitoring in real time a scalar quality characteristic adjusted for by the effect of multivariate functional covariates.
control_charts_sof_pc_real_time

Usage

control_charts_sof_pc_real_time(
  mod_list,
  y_test,
  mfdobj_x_test,
  mfdobj_x_tuning = NULL,
  alpha = list(T2 = 0.0125, spe = 0.0125, y = 0.025),
  limits = "standard",
  seed,
  nfold = NULL,
  ncores = 1
)

Arguments

mod_list

A list of lists produced by sof_pc_real_time, containing a list of scalar-on-function linear regression models estimated on functional data each evolving up to an intermediate domain point.

y_test

A numeric vector containing the observations of the scalar response variable in the phase II monitoring data set.

mfdobj_x_test

A list created using get_mfd_df_real_time or get_mfd_list_real_time, denoting a list of functional data objects in the phase II monitoring data set, each evolving up to an intermediate domain point, with observations of the multivariate functional covariates. The length of this list and mod_list must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point.

mfdobj_x_tuning

A list created using get_mfd_df_real_time or get_mfd_list_real_time, denoting a list of functional data objects in the tuning data set (used to estimate control chart limits), each evolving up to an intermediate domain point, with observations of the multivariate functional covariates. The length of this list and mod_list must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point. If NULL, the training data, i.e. the functional covariates in mod_list, are also used as the tuning data set. Default is NULL.

alpha

See control_charts_sof_pc.

limits

See control_charts_sof_pc.

seed

Deprecated: see control_charts_sof_pc.

nfold

See control_charts_sof_pc.

ncores

If you want parallelization, give the number of cores/threads to be used when creating objects separately for different instants.

Value

A list of data.frames each produced by control_charts_sof_pc, corresponding to a given instant.
The `cont_plot` function produces a contribution plot from functional control charts for a given observation of a phase II data set, using ggplot.

### Usage

```r
cont_plot(cclist, id_num, which_plot = c("T2", "spe"), print_id = FALSE)
```

### Arguments

- `cclist`: A data frame produced by `control_charts_pca`, `control_charts_sof_pc`, `regr_cc_fof`, or `regr_cc_sof`.
- `id_num`: An index number giving the observation in the phase II data set to be plotted, i.e. 1 for the first observation, 2 for the second, and so on.
- `which_plot`: A character vector. Each value indicates which contribution you want to plot: "T2" indicates contribution to the Hotelling’s $T^2$ statistic, "spe" indicates contribution to the squared prediction error statistic.

### Description

This function produces a contribution plot from functional control charts for a given observation of a phase II data set, using ggplot.

### Examples

```r
library(funcharts)
data("air")
air1 <- lapply(air, function(x) x[1:8, , drop = FALSE])
air2 <- lapply(air, function(x) x[9:10, , drop = FALSE])
mfdobj_x1_list <- get_mfd_list_real_time(air1[c("CO", "temperature")],
n_basis = 15,
lambda = 1e-2,
k_seq = c(0.5, 1))
mfdobj_x2_list <- get_mfd_list_real_time(air2[c("CO", "temperature")],
n_basis = 15,
lambda = 1e-2,
k_seq = c(0.5, 1))
y1 <- rowMeans(air1$NO2)
y2 <- rowMeans(air2$NO2)
mod_list <- sof_pc_real_time(y1, mfdobj_x1_list)
cclist <- control_charts_sof_pc_real_time(
  mod_list = mod_list,
  y_test = y2,
  mfdobj_x_test = mfdobj_x2_list)
plot_control_charts_real_time(cclist, 1)
```
print_id A logical value, if TRUE, it prints also the id of the observation in the title of the ggplot. Default is FALSE.

Value
A ggplot containing the contributions of functional variables to the monitoring statistics. Each plot is a bar plot, with bars corresponding to contribution values and horizontal black segments denoting corresponding (empirical) upper limits. Bars are coloured by red if contributions exceed their limit.

Examples
library(funcharts)
data("air")
air <- lapply(air, function(x) x[201:300, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates],
                        n_basis = 15,
                        lambda = 1e-2)
y <- rowMeans(air$NO2)
y1 <- y[1:60]
y2 <- y[91:100]
mfdobj_x1 <- mfdobj_x[1:60]
mfdobj_x_tuning <- mfdobj_x[61:90]
mfdobj_x2 <- mfdobj_x[91:100]
mod <- sof_pc(y1, mfdobj_x1)
cclist <- control_charts_sof_pc(mod = mod,
                                 y_test = y2,
                                 mfdobj_x_test = mfdobj_x2,
                                 mfdobj_x_tuning = mfdobj_x_tuning)
get_ooc(cclist)
cont_plot(cclist, 3)

data_sim_mfd Simulate multivariate functional data

Description
Simulate random coefficients and create a multivariate functional data object of class ‘mfd’.

Usage
data_sim_mfd(nobs = 5, nbasis = 5, nvar = 2, seed)

Arguments
nobs Number of functional observations to be simulated.
nbasis Number of basis functions.
nvar Number of functional covariates.
seed Deprecated: use set.seed() before calling the function for reproducibility.
fof_pc

Value

A simulated object of class ‘mfd’.

Examples

library(funcharts)
data_sim_mfd()

Description

Function-on-function linear regression based on principal components. This function performs multivariate functional principal component analysis (MFPCA) to extract multivariate functional principal components from the multivariate functional covariates as well as from the functional response, then it builds a linear regression model of the response scores on the covariate scores. Both functional covariates and response are standardized before the regression. See Centofanti et al. (2021) for additional details.

Usage

fof_pc(
  mfdobj_y,
  mfdobj_x,
  tot_variance_explained_x = 0.95,
  tot_variance_explained_y = 0.95,
  tot_variance_explained_res = 0.95,
  components_x = NULL,
  components_y = NULL,
  type_residuals = "standard"
)

Arguments

mfdobj_y A multivariate functional data object of class mfd denoting the functional response variable. Although it is a multivariate functional data object, it must have only one functional variable.

mfdobj_x A multivariate functional data object of class mfd denoting the functional covariates.

tot_variance_explained_x The minimum fraction of variance that has to be explained by the multivariate functional principal components retained into the MFPCA model fitted on the functional covariates. Default is 0.95.
**tot_variance_explained_y**

The minimum fraction of variance that has to be explained by the multivariate functional principal components retained into the MFPCA model fitted on the functional response. Default is 0.95.

**tot_variance_explained_res**

The minimum fraction of variance that has to be explained by the multivariate functional principal components retained into the MFPCA model fitted on the functional residuals of the functional regression model. Default is 0.95.

**components_x**

A vector of integers with the components over which to project the functional covariates. If NULL, the first components that explain a minimum fraction of variance equal to \( \text{tot_variance_explained}_x \) is selected. If this is not NULL, the criteria to select components are ignored. Default is NULL.

**components_y**

A vector of integers with the components over which to project the functional response. If NULL, the first components that explain a minimum fraction of variance equal to \( \text{tot_variance_explained}_y \) is selected. If this is not NULL, the criteria to select components are ignored. Default is NULL.

**type_residuals**

A character value that can be "standard" or "studentized". If "standard", the MFPCA on functional residuals is calculated on the standardized covariates and response. If "studentized", the MFPCA on studentized version of the functional residuals is calculated on the non-standardized covariates and response. See Centofanti et al. (2021) for additional details.

**Value**

A list containing the following arguments:

* mod: an object of class `lm` that is a linear regression model where the response variables are the MFPCA scores of the response variable and the covariates are the MFPCA scores of the functional covariates. `mod$coefficients` contains the matrix of coefficients of the functional regression basis functions,

* beta_fd: a `bi_fd` object containing the bivariate functional regression coefficients \( \beta(s,t) \) estimated with the function-on-function linear regression model,

* fitted.values: a multivariate functional data object of class `mfd` with the fitted values of the functional response observations based on the function-on-function linear regression model,

* residuals_original_scale: a multivariate functional data object of class `mfd` with the functional residuals of the function-on-function linear regression model on the original scale, i.e. they are the difference between `mfdobj_y` and `fitted.values`,

* residuals: a multivariate functional data object of class `mfd` with the functional residuals of the function-on-function linear regression model, standardized or studentized depending on the argument `type_residuals`,

* type_residuals: the same as the provided argument,

* pca_x: an object of class `pca_mfd` obtained by doing MFPCA on the functional covariates,

* pca_y: an object of class `pca_mfd` obtained by doing MFPCA on the functional response,

* pca_res: an object of class `pca_mfd` obtained by doing MFPCA on the functional residuals,

* components_x: a vector of integers with the components selected in the `pca_x` model,
fof_pc_real_time

* components_y: a vector of integers with the components selected in the pca_y model,
* components_res: a vector of integers with the components selected in the pca_res model,
* y_standardized: the standardized functional response obtained doing scale_mfd(mfdobj_y),
* tot_variance_explained_x: the same as the provided argument
* tot_variance_explained_y: the same as the provided argument
* tot_variance_explained_res: the same as the provided argument
* get_studentized_residuals: a function that allows to calculate studentized residuals on new data, given the estimated function-on-function linear regression model.

References


Examples

library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj <- get_mfd_list(air, lambda = 1e-2)
mfdobj_y <- mfdobj[, "NO2"]
mfdobj_x <- mfdobj[, fun_covariates]
mod <- fof_pc(mfdobj_y, mfdobj_x)

fof_pc_real_time

Get a list of function-on-function linear regression models estimated on functional data each evolving up to an intermediate domain point.

Description

This function produces a list of objects, each of them contains the result of applying fof_pc to a functional response variable and multivariate functional covariates evolved up to an intermediate domain point.

Usage

fof_pc_real_time(  mfdobj_y_list,  mfdobj_x_list,  tot_variance_explained_x = 0.95,  tot_variance_explained_y = 0.95,  tot_variance_explained_res = 0.95,  components_x = NULL,  components_y = NULL),
type_residuals = "standard",
ncores = 1
)

Arguments

mfdobj_y_list A list created using get_mfd_df_real_time or get_mfd_list_real_time, de-
noting a list of functional data objects, each evolving up to an intermediate do-
main point, with observations of the functional response variable.

mfdobj_x_list A list created using get_mfd_df_real_time or get_mfd_list_real_time, de-
noting a list of functional data objects, each evolving up to an intermediate do-
main point, with observations of the multivariate functional covariates.

tot_variance_explained_x
See fof_pc.
tot_variance_explained_y
See fof_pc.
tot_variance_explained_res
See fof_pc.
components_x See fof_pc.
components_y See fof_pc.
type_residuals See fof_pc.
ncores If you want parallelization, give the number of cores/threads to be used when
creating objects separately for different instants.

Value

A list of lists each produced by fof_pc, corresponding to a given instant.

See Also

fof_pc, get_mfd_df_real_time, get_mfd_list_real_time

Examples

library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])

mfdobj_y_list <- get_mfd_list_real_time(air["NO2"],
n_basis = 15,
lambda = 1e-2,
k_seq = c(0.5, 0.75, 1))
mfdobj_x_list <- get_mfd_list_real_time(air[c("CO", "temperature")],
n_basis = 15,
lambda = 1e-2,
k_seq = c(0.5, 0.75, 1))
mod_list <- fof_pc_real_time(mfdobj_y_list, mfdobj_x_list)
package

Description


References


geom_mfd

*Create a geom layer to plot a Multivariate Functional Data Object with ggplot*

Description

Creates a geom layer to plot a Multivariate Functional Data Object with ggplot

Usage

```r
geom_mfd(
  mapping = NULL,
  data = NULL,
  mfdobj,
  stat = "identity",
  position = "identity",
  na.rm = TRUE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE,
  type_mfd = "mfd",
  ...)
```

Arguments

mapping  Set of aesthetic mappings additional to x and y as passed to the function ggplot2::geom_line.
data  A data.frame providing columns to create additional aesthetic mappings. It must contain a column "id" with the replication values as in mfdobj$fdnames[[2]]. If it contains a column "var", this must contain the functional variables as in mfdobj$fdnames[[3]].
mfdobj  A multivariate functional data object of class mfd.
stat  See ggplot2::geom_line.
position  See ggplot2::geom_line.
na.rm  See ggplot2::geom_line.
orientation  See ggplot2::geom_line.
show.legend  See ggplot2::geom_line.
inherit.aes  See ggplot2::geom_line.
type_mfd  A character value equal to "mfd" or "raw". If "mfd", the smoothed functional data are plotted, if "raw", the original discrete data are plotted.
...

Value

A geom_line layer to be added to ggplot2::ggplot() in order to plot mfdobj.

Examples

library(funcharts)
mfdobj <- data_sim_mfd()
ids <- mfdobj$fdnames[[2]]
df <- data.frame(id = ids, first_two_obs = ids %in% c("rep1", "rep2"))

ggplot() +
  geom_mfd(mapping = aes(colour = first_two_obs),
           data = df,
           mfdobj = mfdobj)
Usage

```
get_mfd_array(
  data_array,
  grid = NULL,
  n_basis = 30,
  lambda = NULL,
  lambda_grid = 10^seq(-10, 1, length.out = 10),
  ncores = 1
)
```

Arguments

data_array A three-dimensional array. The first dimension corresponds to argument values, the second to replications, and the third to variables within replications.

grid See `get_mfd_list`.

n_basis See `get_mfd_list`.

lambda See `get_mfd_list`.

lambda_grid See `get_mfd_list`.

ncores Deprecated. See `get_mfd_list`.

Value

An object of class `mfd`. See also `?mfd` for additional details on the multivariate functional data class.

See Also

`get_mfd_list`, `get_mfd_df`

Examples

```
library(funcharts)
data("CanadianWeather")
mfdobj <- get_mfd_array(CanadianWeather$dailyAv[, 1:10, ],
                         lambda = 1e-5)
plot_mfd(mfdobj)
```

---

`get_mfd_array_real_time`  
Get a list of functional data objects each evolving up to an intermediate domain point.
Description

This function produces a list functional data objects, each evolving up to an intermediate domain point, that can be used to estimate models that allow real-time predictions of incomplete functions, from the current functional domain up to the end of the observation, and to build control charts for real-time monitoring.

It calls the function `get_mfd_array` for each domain point.

Usage

```r
get_mfd_array_real_time(
  data_array,
  grid = NULL,
  n_basis = 30,
  lambda = NULL,
  lambda_grid = 10^seq(-10, 1, length.out = 10),
  k_seq = seq(from = 0.25, to = 1, length.out = 10),
  ncores = 1
)
```

Arguments

- `data_array`: See `get_mfd_array`.
- `grid`: See `get_mfd_array`.
- `n_basis`: See `get_mfd_array`.
- `lambda`: See `get_mfd_array`.
- `lambda_grid`: See `get_mfd_array`.
- `k_seq`: A vector of values between 0 and 1, containing the domain points over which functional data are to be evaluated in real time. If the domain is the interval (a,b), for each instant k in the sequence, functions are evaluated in (a,k(b-a)).
- `ncores`: If you want parallelization, give the number of cores/threads to be used when creating mfd objects separately for different instants.

Value

A list of mfd objects as produced by `get_mfd_array`.

See Also

`get_mfd_array`

Examples

```r
library(funcharts)
data("CanadianWeather")
fdobj <- get_mfd_array_real_time(CanadianWeather$dailyAv[, 1:5, 1:2],
  lambda = 1e-2)
```
get_mfd_df

Get Multivariate Functional Data from a data frame

Description

Get Multivariate Functional Data from a data frame

Usage

get_mfd_df(
  dt,
  domain,
  arg,
  id,
  variables,
  n_basis = 30,
  lambda = NULL,
  lambda_grid = 10^seq(-10, 1, length.out = 10),
  ncores = 1
)

Arguments

dt A data.frame containing the discrete data. For each functional variable, a single column, whose name is provided in the argument variables, contains discrete values of that variable for all functional observation. The column indicated by the argument id denotes which is the functional observation in each row. The column indicated by the argument arg gives the argument value at which the discrete values of the functional variables are observed for each row.

domain A numeric vector of length 2 defining the interval over which the functional data object can be evaluated.

arg A character variable, which is the name of the column of the data frame dt giving the argument values at which the functional variables are evaluated for each row.

id A character variable indicating which is the functional observation in each row.

variables A vector of characters of the column names of the data frame dt indicating the functional variables.

n_basis An integer variable specifying the number of basis functions; default value is 30. See details on basis functions.

lambda A non-negative real number. If you want to use a single specified smoothing parameter for all functional data objects in the dataset, this argument is passed to the function fda::fdPar. Default value is NULL, in this case the smoothing parameter is chosen by minimizing the generalized cross-validation (GCV) criterion over the grid of values given by the argument. See details on how smoothing parameters work.
lambda_grid  A vector of non-negative real numbers. If lambda is provided as a single number, this argument is ignored. If lambda is NULL, then this provides the grid of values over which the optimal smoothing parameter is searched. Default value is 10^seq(-10,1,l=20).

ncores  If you want parallelization, give the number of cores/threads to be used when doing GCV separately on all observations.

Details

Basis functions are created with fda::create.bspline.basis(domain, n_basis), i.e. B-spline basis functions of order 4 with equally spaced knots are used to create mfd objects.

The smoothing penalty lambda is provided as fda::fdPar(bs, 2, lambda), where bs is the basis object and 2 indicates that the integrated squared second derivative is penalized.

Rather than having a data frame with long format, i.e. with all functional observations in a single column for each functional variable, if all functional observations are observed on a common equally spaced grid, discrete data may be available in matrix form for each functional variable. In this case, see get_mfd_list.

Value

An object of class mfd. See also ?mfd for additional details on the multivariate functional data class.

See Also

get_mfd_list

Examples

library(funcharts)

x <- seq(1, 10, length = 25)
y11 <- cos(x)
y21 <- cos(2*x)
y12 <- sin(x)
y22 <- sin(2*x)
df <- data.frame(id = factor(rep(1:2, each = length(x))),
    x = rep(x, times = 2),
    y1 = c(y11, y21),
    y2 = c(y12, y22))

mfdobj <- get_mfd_df(dt = df,
    domain = c(1, 10),
    arg = "x",
    id = "id",
    variables = c("y1", "y2"),
    lambda = 1e-5)
**get_mfd_df_real_time**  
Get a list of functional data objects each evolving up to an intermediate domain point.

**Description**

This function produces a list functional data objects, each evolving up to an intermediate domain point, that can be used to estimate models that allow real-time predictions of incomplete functions, from the current functional domain up to the end of the observation, and to build control charts for real-time monitoring.

It calls the function `get_mfd_df` for each domain point.

**Usage**

```r
get_mfd_df_real_time(
    dt,
    domain,
    arg,
    id,
    variables,
    n_basis = 30,
    lambda = NULL,
    lambda_grid = 10^seq(-10, 1, length.out = 10),
    k_seq = seq(from = 0.25, to = 1, length.out = 10),
    ncores = 1
)
```

**Arguments**

- `dt`: See `get_mfd_df`.
- `domain`: See `get_mfd_df`.
- `arg`: See `get_mfd_df`.
- `id`: See `get_mfd_df`.
- `variables`: See `get_mfd_df`.
- `n_basis`: See `get_mfd_df`.
- `lambda`: See `get_mfd_df`.
- `lambda_grid`: See `get_mfd_df`.
- `k_seq`: A vector of values between 0 and 1, containing the domain points over which functional data are to be evaluated in real time. If the domain is the interval (a,b), for each instant k in the sequence, functions are evaluated in (a,k(b-a)).
- `ncores`: If you want parallelization, give the number of cores/threads to be used when creating mfd objects separately for different instants.
get_mfd_fd

Convert a fd object into a Multivariate Functional Data object.

Description

Convert a fd object into a Multivariate Functional Data object.

Usage

get_mfd_fd(fobj)

Arguments

fobj An object of class fd.

Value

An object of class mfd. See also ?mfd for additional details on the multivariate functional data class.
**get_mfd_list**

See Also

mfd

Examples

library(funcharts)
bs <- create.bspline.basis(nbasis = 10)
fdobj <- fd(coef = 1:10, basisobj = bs)
mfdobj <- get_mfd_fd(fdobj)

**Description**

Get Multivariate Functional Data from a list of matrices

**Usage**

get_mfd_list(
  data_list,
  grid = NULL,
  n_basis = 30,
  lambda = NULL,
  lambda_grid = 10^seq(-10, 1, length.out = 10),
  ncores = 1
)

**Arguments**

data_list A named list of matrices. Names of the elements in the list denote the functional variable names. Each matrix in the list corresponds to a functional variable. All matrices must have the same dimension, where the number of rows corresponds to replications, while the number of columns corresponds to the argument values at which functions are evaluated.

grid A numeric vector, containing the argument values at which functions are evaluated. Its length must be equal to the number of columns in each matrix in data_list. Default is NULL, in this case a vector equally spaced numbers between 0 and 1 is created, with as many numbers as the number of columns in each matrix in data_list.

n_basis An integer variable specifying the number of basis functions; default value is 30. See details on basis functions.

lambda A non-negative real number. If you want to use a single specified smoothing parameter for all functional data objects in the dataset, this argument is passed to the function fda:::fdPar. Default value is NULL, in this case the smoothing parameter is chosen by minimizing the generalized cross-validation (GCV) criterion over the grid of values given by the argument. See details on how smoothing parameters work.
get_mfd_list_real_time

Get a list of functional data objects each evolving up to an intermediate domain point.

Description

This function produces a list functional data objects, each evolving up to an intermediate domain point, that can be used to estimate models that allow real-time predictions of incomplete functions, from the current functional domain up to the end of the observation, and to build control charts for real-time monitoring.

It calls the function get_mfd_list for each domain point.

lambda_grid  A vector of non-negative real numbers. If lambda is provided as a single number, this argument is ignored. If lambda is NULL, then this provides the grid of values over which the optimal smoothing parameter is searched. Default value is $10^{\text{seq}(-10,1,l=20)}$.

ncores  Deprecated.

Details

Basis functions are created with fda::create.bspline.basis(domain, n_basis), i.e. B-spline basis functions of order 4 with equally spaced knots are used to create mfd objects.

The smoothing penalty lambda is provided as fda::fdPar(bs, 2, lambda), where bs is the basis object and 2 indicates that the integrated squared second derivative is penalized.

Rather than having a list of matrices, you may have a data frame with long format, i.e. with all functional observations in a single column for each functional variable. In this case, see get_mfd_df.

Value

An object of class mfd. See also mfd for additional details on the multivariate functional data class.

See Also

mfd, get_mfd_list, get_mfd_array

Examples

library(funcharts)
data("air")
# Only take first 5 multivariate functional observations
# and only two variables from air
air_small <- lapply(air[,c("NO2", "CO")], function(x) x[1:5, ])
mdobj <- get_mfd_list(data_list = air_small)
get_mfd_list_real_time

Usage

get_mfd_list_real_time(
  data_list,
  grid = NULL,
  n_basis = 30,
  lambda = NULL,
  lambda_grid = 10^seq(-10, 1, length.out = 10),
  k_seq = seq(from = 0.2, to = 1, by = 0.1),
  ncores = 1
)

Arguments

data_list See get_mfd_list.
grid See get_mfd_list.
n_basis See get_mfd_list.
lambda See get_mfd_list.
lambda_grid See get_mfd_df.
k_seq A vector of values between 0 and 1, containing the domain points over which functional data are to be evaluated in real time. If the domain is the interval (a,b), for each instant k in the sequence, functions are evaluated in (a,a+k(b-a)).
ncores If you want parallelization, give the number of cores/threads to be used when creating mfd objects separately for different instants.

Value

A list of mfd objects as produced by get_mfd_list.

See Also

get_mfd_list

Examples

library(funcharts)
data("air")
# Only take first 5 multivariate functional observations from air
air_small <- lapply(air, function(x) x[1:5, ])
# Consider only 3 domain points: 0.5, 0.75, 1
mfdobj <- get_mfd_list_real_time(data_list = air_small,
                                  lambda = 1e-2,
                                  k_seq = c(0.5, 0.75, 1))
get_ooc

Get out of control observations from control charts

Description

Get out of control observations from control charts

Usage

get_ooc(cclist)

Arguments

cclist  A data.frame produced by control_charts_pca, control_charts_sof_pc, regr_cc_fof, or regr_cc_sof.

Value

A data.frame with the same number of rows as cclist, and the same number of columns apart from the columns indicating control chart limits. Each value is TRUE if the corresponding observation is in control and FALSE otherwise.

Examples

library(funcharts)
data("air")
air <- lapply(air, function(x) x[201:300, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates],
n_basis = 15,
lambda = 1e-2)
y <- rowMeans(air$NO2)
y1 <- y[1:60]
y2 <- y[91:100]
mfdobj_x1 <- mfdobj_x[1:60]
mfdobj_x_tuning <- mfdobj_x[61:90]
mfdobj_x2 <- mfdobj_x[91:100]
mod <- sof_pc(y1, mfdobj_x1)
cclist <- control_charts_sof_pc(mod = mod,
y_test = y2,
mfdobj_x_test = mfdobj_x2,
mfdobj_x_tuning = mfdobj_x_tuning)
get_ooc(cclist)
get_sof_pc_outliers

Get possible outliers of a training data set of a scalar-on-function regression model.

Description
Get possible outliers of a training data set of a scalar-on-function regression model. It sets the training data set also as tuning data set for the calculation of control chart limits, and as phase II data set to compare monitoring statistics against the limits and identify possible outliers. This is only an empirical approach. It is advised to use methods appropriately designed for phase I monitoring to identify outliers.

Usage
get_sof_pc_outliers(y, mfdobj)

Arguments
y
A numeric vector containing the observations of the scalar response variable.
mfdobj
A multivariate functional data object of class mfd denoting the functional covariates.

Value
A character vector with the ids of functional observations signaled as possibly anomalous.

Examples
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates], lambda = 1e-2)
y <- rowMeans(air$NO2)
get_sof_pc_outliers(y, mfdobj_x)

inprod_mfd
Inner products of functional data contained in mfd objects.

Description
Inner products of functional data contained in mfd objects.

Usage
inprod_mfd(mfdobj1, mfdobj2 = NULL)
Arguments

mfdobj1
A multivariate functional data object of class mfd.

mfdobj2
A multivariate functional data object of class mfd. It must have the same functional variables as mfdobj1. If NULL, it is equal to mfdobj1.

Details

Note that $L^2$ inner products are not calculated for couples of functional data from different functional variables. This function is needed to calculate the inner product in the product Hilbert space in the case of multivariate functional data, which for each observation is the sum of the $L^2$ inner products obtained for each functional variable.

Value

A three-dimensional array of $L^2$ inner products. The first dimension is the number of functions in argument mfdobj1, the second dimension is the same thing for argument mfdobj2, the third dimension is the number of functional variables. If you sum values over the third dimension, you get a matrix of inner products in the product Hilbert space of multivariate functional data.

Examples

```r
library(funcharts)
set.seed(123)
mfdobj1 <- data_sim_mfd()
mfdobj2 <- data_sim_mfd()
inprod_mfd(mfdobj1)
inprod_mfd(mfdobj1, mfdobj2)
```

Description

Inner product of two multivariate functional data objects, for each observation

Usage

```
inprod_mfd_diag(mfdobj1, mfdobj2 = NULL)
```

Arguments

- mfdobj1: A multivariate functional data object of class mfd.
- mfdobj2: A multivariate functional data object of class mfd, with the same number of functional variables and observations as mfdobj1. If NULL, then mfdobj2=mfdobj1. Default is NULL.
Value
It calculates the inner product of two multivariate functional data objects. The main function `inprod` of the package `fda` calculates inner products among all possible couples of observations. This means that, if `mfdobj1` has \( n_1 \) observations and `mfdobj2` has \( n_2 \) observations, then for each variable \( n_1 \times n_2 \) inner products are calculated. However, often one is interested only in calculating the \( n \) inner products between the \( n \) observations of `mfdobj1` and the corresponding \( n \) observations of `mfdobj2`. This function provides this "diagonal" inner products only, saving a lot of computation with respect to using `fda::inprod` and then extracting the diagonal elements. Note that the code of this function calls a modified version of `fda::inprod()`.

Examples
```r
mfdobj <- data_sim_mfd()
inprod_mfd_diag(mfdobj)
```

---

### is.mfd

**Confirm Object has Class mfd**

**Description**
Check that an argument is a multivariate functional data object of class `mfd`.

**Usage**
```r
is.mfd(mfdobj)
```

**Arguments**
- `mfdobj` An object to be checked.

**Value**
a logical value: TRUE if the class is correct, FALSE otherwise.

---

### mfd

**Define a Multivariate Functional Data Object**

**Description**
This is the constructor function for objects of the `mfd` class. It is a wrapper to `fda::fd`, but it forces the `coef` argument to be a three-dimensional array of coefficients even if the functional data is univariate. Moreover, it allows to include the original raw data from which you get the smooth functional data.
Usage

mfd(coef, basisobj, fdnames = NULL, raw = NULL, id_var = NULL, B = NULL)

Arguments

coef A three-dimensional array of coefficients:
* the first dimension corresponds to basis functions.
* the second dimension corresponds to the number of multivariate functional
  observations.
* the third dimension corresponds to variables.
basisobj A functional basis object defining the basis, as provided to fda::fd, but there is
  no default.
fdnames A list of length 3, each member being a string vector containing labels for the
  levels of the corresponding dimension of the discrete data.
The first dimension is for a single character indicating the argument values, i.e.
the variable on the functional domain.
The second is for replications, i.e. it denotes the functional observations.
The third is for functional variables’ names.
raw A data frame containing the original discrete data. Default is NULL, however,
if provided, it must contain:
a column (indicated by the id_var argument) denoting the functional observations,
which must correspond to values in fdnames[[2]].
a column named as fdnames[[1]], returning the argument values of each function
as many columns as the functional variables, named as in fdnames[[3]], contain-
ing the discrete functional values for each variable.

id_var A single character value indicating the column in the raw argument containing
the functional observations (as in fdnames[[2]]), default is NULL.
B A matrix with the inner products of the basis functions. If NULL, it is calculated
from the basis object provided. Default is NULL.

Details

To check that an object is of this class, use function is.mfd.

Value

A multivariate functional data object (i.e., having class mfd), which is a list with components named
coefs, basis, and fdnames, as for class fd, with possibly in addition the components raw and
id_var.

References

New York.

New York.
Examples

```r
library(funcharts)
set.seed(0)
nobs <- 5
nbasis <- 10
nvar <- 2
coeff <- array(rnorm(nobs * nbasis * nvar), dim = c(nbasis, nobs, nvar))
bs <- create.bspline.basis(rangeval = c(0, 1), nbasis = nbasis)
mfdobj <- mfd(coef = coeff, basisobj = bs)
plot_mfd(mfdobj)
```

---

**norm.mfd**

*Norm of Multivariate Functional Data*

**Description**

Norm of multivariate functional data contained in a mfd object.

**Usage**

```r
norm.mfd(mfdobj)
```

**Arguments**

- `mfdobj` A multivariate functional data object of class mfd.

**Value**

A vector of length equal to the number of replications in mfdobj, containing the norm of each multivariate functional observation in the product Hilbert space, i.e. the sum of $L^2$ norms for each functional variable.

**Examples**

```r
library(funcharts)
mfdobj <- data_sim_mfd()
norm.mfd(mfdobj)
```
Multivariate functional principal components analysis (MFPCA) performed on an object of class \textit{mfd}. It is a wrapper to \texttt{fda::pca.fd}, providing some additional arguments.

\textbf{Usage}

\begin{verbatim}
pca_mfd(mfdobj, scale = TRUE, nharm = 20)
\end{verbatim}

\textbf{Arguments}

- \texttt{mfdobj} A multivariate functional data object of class \textit{mfd}.
- \texttt{scale} If \texttt{TRUE}, it scales data before doing MFPCA using \texttt{scale_mfd}. Default is \texttt{TRUE}.
- \texttt{nharm} Number of multivariate functional principal components to be calculated. Default is 20.

\textbf{Value}

Modified \texttt{pca.fd} object, with multivariate functional principal component scores summed over variables (\texttt{fda::pca.fd} returns an array of scores when providing a multivariate functional data object). Moreover, the multivariate functional principal components given in harmonics are converted to the \textit{mfd} class.

\textbf{See Also}

\texttt{scale_mfd}

\textbf{Examples}

\begin{verbatim}
library(funcharts)
mfdobj <- data_sim_mfd()
pca_obj <- pca_mfd(mfdobj)
plot_pca_mfd(pca_obj)
\end{verbatim}
pca_mfd_real_time

Description

This function produces a list of objects, each of them contains the result of applying `pca_mfd` to a multivariate functional data object evolved up to an intermediate domain point.

Usage

```r
pca_mfd_real_time(mfdobj_list, scale = TRUE, nharm = 20, ncores = 1)
```

Arguments

- **mfdobj_list**: A list created using `get_mfd_df_real_time` or `get_mfd_list_real_time`, denoting a list of functional data objects, each evolving up to an intermediate domain point, with observations of the multivariate functional data.
- **scale**: See `pca_mfd`.
- **nharm**: See `pca_mfd`.
- **ncores**: If you want parallelization, give the number of cores/threads to be used when creating objects separately for different instants.

Value

A list of lists each produced by `pca_mfd`, corresponding to a given instant.

See Also

- `pca_mfd`

Examples

```r
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])
mfdobj_list <- get_mfd_list_real_time(air[c("CO", "temperature")],
                                      n_basis = 15,
                                      lambda = 1e-2,
                                      k_seq = seq(0.25, 1, length = 5))
mod_list <- pca_mfd_real_time(mfdobj_list)
```
**plot_bifd**  
*Plot a Bivariate Functional Data Object.*

**Description**

Plot an object of class `bifd` using `ggplot2` and `geom_tile`. The object must contain only one single functional replication.

**Usage**

```r
plot_bifd(bifd_obj)
```

**Arguments**

- `bifd_obj` A bivariate functional data object of class `bifd`, containing one single replication.

**Value**

A `ggplot` with a `geom_tile` layer providing a plot of the bivariate functional data object as a heatmap.

**Examples**

```r
library(funcharts)
mfdobj <- data_sim_mfd(nobs = 1)
tp <- tensor_product_mfd(mfdobj)
plot_bifd(tp)
```

---

**plot_bootstrap_sof_pc**  
*Plot bootstrapped estimates of the scalar-on-function regression coefficient*

**Description**

Plot bootstrapped estimates of the scalar-on-function regression coefficient for empirical uncertainty quantification. For each iteration, a data set is sampled with replacement from the training data used to fit the model, and the regression coefficient is estimated.

**Usage**

```r
plot_bootstrap_sof_pc(mod, nboot = 25, ncores = 1)
```
### Arguments

- **mod**
  A list obtained as output from *sof_pc*, i.e. a fitted scalar-on-function linear regression model.
- **nboot**
  Number of bootstrap replicates
- **ncores**
  If you want estimate the bootstrap replicates in parallel, give the number of cores/threads.

### Value

A ggplot showing several bootstrap replicates of the multivariate functional coefficients estimated fitting the scalar-on-function linear model. Gray lines indicate the different bootstrap estimates, the black line indicate the estimate on the entire dataset.

### Examples

```r
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates], lambda = 1e-2)
y <- rowMeans(air$NO2)
mod <- sof_pc(y, mfdobj_x)
plot_bootstrap_sof_pc(mod, nboot = 5)
```

---

### Description

This function takes as input a data frame produced with functions such as *control_charts_pca* and *control_charts_sof_pc* and produces a ggplot with the desired control charts, i.e. it plots a point for each observation in the phase II data set against the corresponding control limits.

### Usage

```r
plot_control_charts(cclist)
```

### Arguments

- **cclist**
  A data.frame produced by *control_charts_pca*, *control_charts_sof_pc* *regr_cc_fof*, or *regr_cc_sof*.

### Details

Out-of-control points are signaled by colouring them in red.
Value

A ggplot with the functional control charts.

Examples

```r
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:100, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates],
  n_basis = 15,
  lambda = 1e-2)
mfdobj_y <- get_mfd_list(air["NO2"],
  n_basis = 15,
  lambda = 1e-2)
mfdobj_y1 <- mfdobj_y[1:60]
mfdobj_y_tuning <- mfdobj_y[61:90]
mfdobj_x1 <- mfdobj_x[1:60]
mfdobj_x_tuning <- mfdobj_x[61:100]
mod_fof <- fof_pc(mfdobj_y1, mfdobj_x1)
cclist <- regr_cc_fof(mod_fof,
  mfdobj_y_new = mfdobj_y2,
  mfdobj_x_new = mfdobj_x2,
  mfdobj_y_tuning = NULL,
  mfdobj_x_tuning = NULL)
plot_control_charts(cclist)
```

Description

This function produces a ggplot with the desired real-time control charts. It takes as input a list of data frames, produced with functions such as `regr_cc_fof_real_time` and `control_charts_sof_pc_real_time`, and the id of the observations for which real-time control charts are desired to be plotted. For each control chart, the solid line corresponds to the profile of the monitoring statistic and it is compared against control limits plotted as dashed lines. If a line is outside its limits it is coloured in red.

Usage

```r
plot_control_charts_real_time(cclist, id_num)
```
**plot_mfd**

Plot a Multivariate Functional Data Object.

**Description**

Plot an object of class mfd using ggplot2.

**Arguments**

- **cclist**: A list of data frames, produced with functions such as `regr_cc_fof_real_time` and `control_charts_sof_pc_real_time`.
- **id_num**: An index number giving the observation in the phase II data set to be plotted, i.e. 1 for the first observation, 2 for the second, and so on.

**Details**

If the line, representing the profile of the monitoring statistic over the functional domain, is out-of-control, then it is coloured in red.

**Value**

A ggplot with the real-time functional control charts.

**See Also**

`regr_cc_fof_real_time`, `control_charts_sof_pc_real_time`

**Examples**

```r
library(funcharts)

data("air")

air1 <- lapply(air, function(x) x[1:8, , drop = FALSE])
air2 <- lapply(air, function(x) x[9:10, , drop = FALSE])

mfdobj_x1_list <- get_mfd_list_real_time(air1[c("CO", "temperature")],
               n_basis = 15,
               lambda = 1e-2,
               k_seq = c(0.5, 1))

mfdobj_x2_list <- get_mfd_list_real_time(air2[c("CO", "temperature")],
               n_basis = 15,
               lambda = 1e-2,
               k_seq = c(0.5, 1))

y1 <- rowMeans(air1$NO2)
y2 <- rowMeans(air2$NO2)

mod_list <- sof_pc_real_time(y1, mfdobj_x1_list)
cclist <- control_charts_sof_pc_real_time(
  mod_list = mod_list,
  y_test = y2,
  mfdobj_x_test = mfdobj_x2_list)

plot_control_charts_real_time(cclist, 1)
```
plot_mfd(mfdobj)

Arguments

mfdobj A multivariate functional data object of class mfd.

Value

an object of class ggplot, created using ggplot() + geom_mfd(mfdobj = mfdobj).

See Also

geom_mfd

Examples

library(funcharts)
mfdobj <- data_sim_mfd()plot_mfd(mfdobj)

plot_mon(cclist, fd_train, fd_test, print_id = FALSE)

Arguments

cclist A data.frame produced by control_charts_pca, control_charts_sof_pc, regr_cc_fof, or regr_cc_sof.
fd_train An object of class mfd containing the training data set of the functional variables. They are plotted in gray in the background.
fd_test An object of class mfd containing the phase II data set of the functional variables to be monitored. They are coloured in black or red on the foreground.
print_id A logical value, if TRUE, it prints also the id of the observation in the title of the ggplot.

Description

This function plots selected functions in a phase_II monitoring data set against the corresponding training data set to be compared.

Usage

plot_mon(cclist, fd_train, fd_test, print_id = FALSE)

Plot multivariate functional object over the training data set
Value

A ggplot of the multivariate functional data. In particular, the multivariate functional data given in `fd_train` are plotted on the background in gray, while the multivariate functional data given in `fd_test` are plotted on the foreground, the colour of each curve is black or red depending on if that curve was signal as anomalous by at least a contribution plot.

Examples

```r
library(funcharts)
data("air")
air <- lapply(air, function(x) x[201:300, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[, fun_covariates],
    n_basis = 15,
    lambda = 1e-2)

y <- rowMeans(air$NO2)
y1 <- y[1:60]
y2 <- y[91:100]
mfdobj_x1 <- mfdobj_x[1:60]
mfdobj_x_tuning <- mfdobj_x[61:90]
mfdobj_x2 <- mfdobj_x[91:100]
mod <- sof_pc(y1, mfdobj_x1)
cclist <- control_charts_sof_pc(mod = mod,
    y_test = y2,
    mfdobj_x_test = mfdobj_x2,
    mfdobj_x_tuning = mfdobj_x_tuning)
plot_control_charts(cclist)
cont_plot(cclist, 3)
plot_mon(cclist, fd_train = mfdobj_x1, fd_test = mfdobj_x2[3])
```

---

**plot_pca_mfd**  
*Plot the harmonics of a pca_mfd object*

**Description**

Plot the harmonics of a pca_mfd object

**Usage**

```r
plot_pca_mfd(pca, harm = 0, scaled = FALSE)
```

**Arguments**

- **pca**: A fitted multivariate functional principal component analysis (MFPCA) object of class pca_mfd.
- **harm**: A vector of integers with the harmonics to plot. If 0, all harmonics are plotted. Default is 0.
scaled
If TRUE, eigenfunctions are multiplied by the square root of the corresponding eigenvalues, if FALSE they are not scaled and all have unit norm. Default is FALSE.

Value
A ggplot of the harmonics-multivariate functional principal components contained in the object pca.

Examples
library(funcharts)
mfdobj <- data_sim_mfd()
pca_obj <- pca_mfd(mfdobj)
plot_pca_mfd(pca_obj)

predict_fof_pc
Use a function-on-function linear regression model for prediction

Description
Predict new observations of the functional response variable and calculate the corresponding prediction error (and their standardized or studentized version) given new observations of functional covariates and a fitted function-on-function linear regression model.

Usage
predict_fof_pc(object, mfdobj_y_new, mfdobj_x_new)

Arguments
object A list obtained as output from fof_pc, i.e., a fitted function-on-function linear regression model.
mfdobj_y_new An object of class mfd containing new observations of the functional response.
mfdobj_x_new An object of class mfd containing new observations of the functional covariates.

Value
A list of mfd objects. It contains:
* pred_error: the prediction error of the standardized functional response variable,
* pred_error_original_scale: the prediction error of the functional response variable on the original scale,
* y_hat_new: the prediction of the functional response observations on the original scale,
* y_z_new: the standardized version of the functional response observations provided in mfdobj_y_new,
* y_hat_z_new: the prediction of the functional response observations on the standardized/studentized scale.
Reference


Examples

library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates], lambda = 1e-2)
mfdobj_y <- get_mfd_list(air["NO2"], lambda = 1e-2)
mod <- fof_pc(mfdobj_y, mfdobj_x)
predict_fof_pc(mod,
    mfdobj_y_new = mfdobj_y,
    mfdobj_x_new = mfdobj_x)

predict_sof_pc

Use a scalar-on-function linear regression model for prediction

Description

Predict new observations of the scalar response variable and calculate the corresponding prediction error, with prediction interval limits, given new observations of functional covariates and a fitted scalar-on-function linear regression model

Usage

predict_sof_pc(object, newdata = NULL, alpha = 0.05)

Arguments

object

A list obtained as output from sof_pc, i.e. a fitted scalar-on-function linear regression model.

newdata

An object of class mfd containing new observations of the functional covariates. If NULL, it is set as the functional covariates data used for model fitting.

alpha

A numeric value indicating the Type I error for the regression control chart and such that this function returns the 1-alpha prediction interval on the response. Default is 0.05.

Value

A data.frame with as many rows as the number of functional replications in newdata, with the following columns:

* fit: the predictions of the response variable corresponding to new_data,
* lwr: lower limit of the 1-alpha prediction interval on the response,
* upr: upper limit of the 1-alpha prediction interval on the response.
Examples

```r
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates], lambda = 1e-2)
y <- rowMeans(air$NO2)
mod <- sof_pc(y, mfdobj_x)
predict_sof_pc(mod)
```

**rbind_mfd**  
Binding replications of two Multivariate Functional Data Objects

**Description**

Bind replications of two Multivariate Functional Data Objects

**Usage**

```r
rbind_mfd(mfdobj1, mfdobj2)
```

**Arguments**

- `mfdobj1`: An object of class `mfd`, with the same variables of `mfdobj2` and different replication names with respect to `mfdobj2`.
- `mfdobj2`: An object of class `mfd`, with the same variables of `mfdobj1`, and different replication names with respect to `mfdobj1`.

**Value**

An object of class `mfd`, whose variables are the same of `mfdobj1` and `mfdobj2` and whose replications are the union of the replications in `mfdobj1` and `mfdobj2`.

**Examples**

```r
library(funcharts)
mfdobj1 <- data_sim_mfd(nvar = 3, nobss = 4)
mfdobj2 <- data_sim_mfd(nvar = 3, nobss = 5)
dimnames(mfdobj2$coefs[[2]] <-
  mfdobj2$fdnames[[2]] <-
c("rep11", "rep12", "rep13", "rep14", "rep15")
mfdobj_rbind <- rbind_mfd(mfdobj1, mfdobj2)
plot_mfd(mfdobj_rbind)
```
Description

It builds a data frame needed to plot the Functional Regression Control Chart introduced in Centofanti et al. (2021), for monitoring a functional quality characteristic adjusted for by the effect of multivariate functional covariates, based on a fitted function-on-function linear regression model. The training data have already been used to fit the model. An optional tuning data set can be provided that is used to estimate the control chart limits. A phase II data set contains the observations to be monitored with the control charts.

Usage

```r
regr_cc_fof(
  object,
  mfdobj_y_new,
  mfdobj_x_new,
  mfdobj_y_tuning = NULL,
  mfdobj_x_tuning = NULL,
  alpha = list(T2 = 0.025, spe = 0.025)
)
```

Arguments

- **object**: A list obtained as output from `fof_pc`, i.e. a fitted function-on-function linear regression model.
- **mfdobj_y_new**: An object of class `mfd` containing the phase II data set of the functional response observations to be monitored.
- **mfdobj_x_new**: An object of class `mfd` containing the phase II data set of the functional covariates observations to be monitored.
- **mfdobj_y_tuning**: An object of class `mfd` containing the tuning data set of the functional response observations, used to estimate the control chart limits. If NULL, the training data, i.e. the data used to fit the function-on-function linear regression model, are also used as the tuning data set, i.e. `mfdobj_y_tuning=object$pca_y$data`. Default is NULL.
- **mfdobj_x_tuning**: An object of class `mfd` containing the tuning data set of the functional covariates observations, used to estimate the control chart limits. If NULL, the training data, i.e. the data used to fit the function-on-function linear regression model, are also used as the tuning data set, i.e. `mfdobj_x_tuning=object$pca_x$data`. Default is NULL.
- **alpha**: A named list with two elements, named T2 and spe, respectively, each containing the desired Type I error probability of the corresponding control chart.
that at the moment you have to take into account manually the family-wise error rate and adjust the two values accordingly. See Centofanti et al. (2021) for additional details. Default value is `list(T2 = 0.025, spe = 0.025).

Value

A `data.frame` containing the output of the function `control_charts_pca` applied to the prediction errors.

References


See Also

`control_charts_pca`

Examples

```r
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:100, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[, fun_covariates],
                        n_basis = 15,
                        lambda = 1e-2)
mfdobj_y <- get_mfd_list(air["NO2"],
                        n_basis = 15,
                        lambda = 1e-2)
mfdobj_y1 <- mfdobj_y[1:60]
mfdobj_y_tuning <- mfdobj_y[61:90]
mfdobj_y2 <- mfdobj_y[91:100]
mfdobj_x1 <- mfdobj_x[1:60]
mfdobj_x_tuning <- mfdobj_x[61:90]
mfdobj_x2 <- mfdobj_x[91:100]
mod_fof <- fof_pc(mfdobj_y1, mfdobj_x1)
cclist <- regr_cc_fof(mod_fof,
                       mfdobj_y_new = mfdobj_y2,
                       mfdobj_x_new = mfdobj_x2,
                       mfdobj_y_tuning = NULL,
                       mfdobj_x_tuning = NULL)
plot_control_charts(cclist)
```
**Description**

This function produces a list of data frames, each of them is produced by `regr_cc_fof` and is needed to plot control charts for monitoring in real time a functional quality characteristic adjusted for by the effect of multivariate functional covariates.

**Usage**

```r
regr_cc_fof_real_time(
  mod_list,
  mfdobj_y_new_list,
  mfdobj_x_new_list,
  mfdobj_y_tuning_list = NULL,
  mfdobj_x_tuning_list = NULL,
  alpha = list(T2 = 0.025, spe = 0.025),
  ncores = 1
)
```

**Arguments**

- `mod_list` A list of lists produced by `fof_pc_real_time`, containing a list of function-on-function linear regression models estimated on functional data each evolving up to an intermediate domain point.

- `mfdobj_y_new_list` A list created using `get_mfd_df_real_time` or `get_mfd_list_real_time`, denoting a list of functional data objects in the phase II monitoring data set, each evolving up to an intermediate domain point, with observations of the functional response variable. The length of this list and `mod_list` must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point.

- `mfdobj_x_new_list` A list created using `get_mfd_df_real_time` or `get_mfd_list_real_time`, denoting a list of functional data objects in the phase II monitoring data set, each evolving up to an intermediate domain point, with observations of the multivariate functional covariates. The length of this list and `mod_list` must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point.

- `mfdobj_y_tuning_list` A list created using `get_mfd_df_real_time` or `get_mfd_list_real_time`, denoting a list of functional data objects in the tuning data set (used to estimate control chart limits), each evolving up to an intermediate domain point, with observations of the functional response variable. The length of this list and `mod_list` must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point.
correspond to the same intermediate domain point. If NULL, the training data, i.e. the functional response in \texttt{mod_list}, is also used as the tuning data set. Default is NULL.

\texttt{mfdobj\_x\_tuning\_list}

A list created using \texttt{get\_mfd\_df\_real\_time} or \texttt{get\_mfd\_list\_real\_time}, denoting a list of functional data objects in the tuning data set (used to estimate control chart limits), each evolving up to an intermediate domain point, with observations of the multivariate functional covariates. The length of this list and \texttt{mod_list} must be equal, and their elements in the same position in the list must correspond to the same intermediate domain point. If NULL, the training data, i.e. the functional covariates in \texttt{mod_list}, are also used as the tuning data set. Default is NULL.

\texttt{alpha}

See \texttt{regr\_cc\_fof}.

\texttt{ncores}

If you want parallelization, give the number of cores/threads to be used when creating objects separately for different instants.

\textbf{Value}

A list of data frames each produced by \texttt{regr\_cc\_fof}, corresponding to a given instant.

\textbf{See Also}

\texttt{fof\_pc\_real\_time, regr\_cc\_fof}

\textbf{Examples}

```r
library(funcharts)
data("air")
air1 <- lapply(air, function(x) x[1:8, , drop = FALSE])
air2 <- lapply(air, function(x) x[9:10, , drop = FALSE])
mfdobj\_x1\_list <- get\_mfd\_list\_real\_time(air1\[c("CO", "temperature")],
                      n\_basis = 15,
                      lambda = 1e-2,
                      k\_seq = c(0.5, 1))
mfdobj\_x2\_list <- get\_mfd\_list\_real\_time(air2\[c("CO", "temperature")],
                      n\_basis = 15,
                      lambda = 1e-2,
                      k\_seq = c(0.5, 1))
mfdobj\_y1\_list <- get\_mfd\_list\_real\_time(air1["NO2"],
                      n\_basis = 15,
                      lambda = 1e-2,
                      k\_seq = c(0.5, 1))
mfdobj\_y2\_list <- get\_mfd\_list\_real\_time(air2["NO2"],
                      n\_basis = 15,
                      lambda = 1e-2,
                      k\_seq = c(0.5, 1))
mod\_list <- fof\_pc\_real\_time(mfdobj\_y1\_list, mfdobj\_x1\_list)
cclist <- regr\_cc\_fof\_real\_time(
  mod\_list = mod\_list,
  mfdobj\_y\_new\_list = mfdobj\_y2\_list,
  mfdobj\_x\_new\_list = mfdobj\_x2\_list)
```
`regr_cc_sof` 51

```r
gl_regr_control_charts(cclist, 1)
```

---

### Description

This function builds a data frame needed to plot the scalar-on-function regression control chart, based on a fitted function-on-function linear regression model and proposed in Capezza et al. (2020) together with the Hotelling’s T^2 and squared prediction error control charts. The training data have already been used to fit the model. A tuning data set can be provided that is used to estimate the control chart limits. A phase II data set contains the observations to be monitored with the built control charts.

### Usage

```r
regr_cc_sof(object, y_new, mfdobj_x_new, alpha = 0.05)
```

### Arguments

- **object**: A list obtained as output from `sof_pc`, i.e. a fitted scalar-on-function linear regression model.
- **y_new**: A numeric vector containing the observations of the scalar response variable in the phase II data set.
- **mfdobj_x_new**: An object of class `mfd` containing the phase II data set of the functional covariates observations.
- **alpha**: A numeric value indicating the Type I error for the regression control chart and such that this function returns the 1-alpha prediction interval on the response. Default is 0.05.

### Value

A `data.frame` with as many rows as the number of functional replications in `mfdobj_x_new`, with the following columns:

- *y_hat*: the predictions of the response variable corresponding to `mfdobj_x_new`,
- *y*: the same as the argument `y_new` given as input to this function,
- *lwr*: lower limit of the 1-alpha prediction interval on the response,
- *pred_err*: prediction error calculated as `y-y_hat`,
- *pred_err_sup*: upper limit of the 1-alpha prediction interval on the prediction error,
- *pred_err_inf*: lower limit of the 1-alpha prediction interval on the prediction error.
References


Examples

```r
library(funcharts)
air <- lapply(air, function(x) x[1:100, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates],
  n_basis = 15,
  lambda = 1e-2)
y <- rowMeans(air$NO2)
y1 <- y[1:80]
y2 <- y[81:100]
mfdobj_x1 <- mfdobj_x[1:80]
mfdobj_x2 <- mfdobj_x[81:100]
mod <- sof_pc(y1, mfdobj_x1)
cclist <- regr_cc_sof(object = mod,
  y_new = y2,
  mfdobj_x_new = mfdobj_x2)
plot_control_charts(cclist)
```

---

**scale_mfd**

*Standardize Multivariate Functional Data.*

**Description**

Scale multivariate functional data contained in an object of class `mfd` by subtracting the mean function and dividing by the standard deviation function.

**Usage**

```r
scale_mfd(mfdobj, center = TRUE, scale = TRUE)
```

**Arguments**

- `mfdobj` A multivariate functional data object of class `mfd`.
- `center` A logical value, or a `fd` object. When providing a logical value, if `TRUE`, `mfdobj` is centered, i.e., the functional mean function is calculated and subtracted from all observations in `mfdobj`, if `FALSE`, `mfdobj` is not centered. If `center` is a `fd` object, then this function is used as functional mean for centering.
- `scale` A logical value, or a `fd` object. When providing a logical value, if `TRUE`, `mfdobj` is scaled after possible centering, i.e., the functional standard deviation is calculated from all functional observations in `mfdobj` and then the observations are divided by this calculated standard deviation, if `FALSE`, `mfdobj` is not scaled. 


simulate_mfd

scaled. If scale is a fd object, then this function is used as standard deviation function for scaling.

Details

This function has been written to work similarly as the function scale for matrices. When calculated, attributes center and scale are of class fd and have the same structure you get when you use fda::mean.fd and fda::sd.fd.

Value

A standardized object of class mfd, with two attributes, if calculated, center and scale, storing the mean and standard deviation functions used for standardization.

Examples

```r
library(funcharts)
mfdobj <- data_sim_mfd()
mfdobj_scaled <- scale_mfd(mfdobj)
```

---

**simulate_mfd**

**Simulate a data set for funcharts**

**Description**

Function used to simulate a data set to illustrate the use of funcharts. It creates a data set with three functional covariates, a functional response generated as a function of the three functional covariates through a function-on-function linear model, and a scalar response generated as a function of the three functional covariates through a scalar-on-function linear model. This function covers the simulation study in Centofanti et al. (2021) for the function-on-function case and also simulates data in a similar way for the scalar response case. In the default case, the function generates in-control data. Additional arguments can be used to generate additional data that are out of control, with mean shifts according to the scenarios proposed by Centofanti et al. (2021). Each simulated observation of a functional variable consists of a vector of 150 discrete points, equally spaced between 0 and 1, generated with noise.

**Usage**

```r
simulate_mfd(
  nobs = 1000,
  R2 = 0.97,
  seed,
  shift_type_y = "0",
  shift_type_x1 = "0",
  shift_type_x2 = "0",
  shift_type_x3 = "0",
  d_y = 0,
  d_x1 = 0,
```
simulate_mfd

d_x2 = 0,
d_x3 = 0,
d_y_scalar = 0,
save_beta = FALSE
)

Arguments

- **nobs**
  The number of observation to simulate

- **R2**
  The desired coefficient of determination in the regression. In both the scalar and functional response cases, only three values are allowed, i.e. 0.97, 0.86, 0.74. Default is 0.97.

- **seed**
  Deprecated: use `set.seed()` before calling the function for reproducibility.

- **shift_type_y**
  The shift type for the functional response. There are five possibilities: "0" if there is no shift, "A", "B", "C" or "D" for the corresponding shift types shown in Centofanti et al. (2021). Default is "0".

- **shift_type_x1**
  The shift type for the first functional covariate. There are five possibilities: "0" if there is no shift, "A", "B", "C" or "D" for the corresponding shift types shown in Centofanti et al. (2021). Default is "0".

- **shift_type_x2**
  The shift type for the second functional covariate. There are five possibilities: "0" if there is no shift, "A", "B", "C" or "D" for the corresponding shift types shown in Centofanti et al. (2021). Default is "0".

- **shift_type_x3**
  The shift type for the third functional covariate. There are five possibilities: "0" if there is no shift, "A", "B", "C" or "D" for the corresponding shift types shown in Centofanti et al. (2021). Default is "0".

- **d_y**
  A number indicating the severity of the shift type for the functional response. Default is 0.

- **d_x1**
  A number indicating the severity of the shift type for the first functional covariate. Default is 0.

- **d_x2**
  A number indicating the severity of the shift type for the second functional covariate. Default is 0.

- **d_x3**
  A number indicating the severity of the shift type for the third functional covariate. Default is 0.

- **d_y_scalar**
  A number indicating the severity of the shift type for the scalar response. Default is 0.

- **save_beta**
  If TRUE, the true regression coefficients of both the function-on-function and the scalar-on-function models are saved. Default is FALSE.

Value

A list with the following elements:

- X1 is a nobsx150 matrix with the simulated observations of the first functional covariate
- X2 is a nobsx150 matrix with the simulated observations of the second functional covariate
- X3 is a nobsx150 matrix with the simulated observations of the third functional covariate
**Simulate example data for funcharts**

Function used to simulate three data sets to illustrate the use of funcharts. It uses the function `simulate_mfd`, which creates a data set with three functional covariates, a functional response generated as a function of the three functional covariates, and a scalar response generated as a function of the three functional covariates. This function generates three data sets, one for phase I, one for tuning, i.e., to estimate the control chart limits, and one for phase II monitoring. see also `simulate_mfd`.

### Usage

```r
sim_funcharts(nobs1 = 1000, nobs_tun = 1000, nobs2 = 60)
```

### Arguments

- **nobs1**: The number of observation to simulate in phase I. Default is 1000.
- **nobs_tun**: The number of observation to simulate the tuning data set. Default is 1000.
- **nobs2**: The number of observation to simulate in phase II. Default is 60.

### Value

A list with three objects, `datI` contains the phase I data, `datI_tun` contains the tuning data, `datII` contains the phase II data. In the phase II data, the first group of observations are in control, the second group of observations contains a moderate mean shift, while the third group of observations contains a severe mean shift. The shift types are described in the paper from Centofanti et al. (2021).

### References

Scalar-on-function linear regression based on principal components. This function performs multivariate functional principal component analysis (MFPCA) to extract multivariate functional principal components from the multivariate functional covariates, then it builds a linear regression model of a scalar response variable on the covariate scores. Functional covariates are standardized before the regression. See Capezza et al. (2020) for additional details.

Usage

sof_pc(
  y, mfdobj_x, tot_variance_explained = 0.9, selection = "variance", single_min_variance_explained = 0, components = NULL
)

Arguments

descriptions:
  y: A numeric vector containing the observations of the scalar response variable.
  mfdobj_x: A multivariate functional data object of class mfd denoting the functional covariates.
  tot_variance_explained: The minimum fraction of variance that has to be explained by the set of multivariate functional principal components retained into the MFPCA model fitted on the functional covariates. Default is 0.9.
  selection: A character value with one of three possible values:
  - if "variance", the first M multivariate functional principal components are retained into the MFPCA model such that together they explain a fraction of variance greater than tot_variance_explained,
  - if "PRESS", each j-th functional principal component is retained into the MFPCA model if, by adding it to the set of the first j-1 functional principal components, then the predicted residual error sum of squares (PRESS) statistic decreases, and at the same time the fraction of variance explained by that single component is greater than single_min_variance_explained. This criterion is used in Capezza et al. (2020).
  - if "gcv", the criterion is equal as in the previous "PRESS" case, but the "PRESS" statistic is substituted by the generalized cross-validation (GCV) score. Default value is "variance".
single_min_variance_explained
The minimum fraction of variance that has to be explained by each multivariate
functional principal component into the MFPCA model fitted on the functional
covariates such that it is retained into the MFPCA model. Default is 0.

components
A vector of integers with the components over which to project the functional
covariates. If this is not NULL, the criteria to select components are ignored. If
NULL, components are selected according to the criterion defined by selection.
Default is NULL.

Value
a list containing the following arguments:
* mod: an object of class lm that is a linear regression model where the scalar response variable
is y and the covariates are the MFPCA scores of the functional covariates, * mod$coefficients
contains the matrix of coefficients of the functional regression basis functions,
* pca: an object of class pca_mfd obtained by doing MFPCA on the functional covariates,
* beta_fd: an object of class mfd object containing the functional regression coefficient \( \beta(t) \)
estimated with the scalar-on-function linear regression model,
* components: a vector of integers with the components selected in the pca model,
* selection: the same as the provided argument
* single_min_variance_explained: the same as the provided argument
* tot_variance_explained: the same as the provided argument
* gcv: a vector whose j-th element is the GCV score obtained when retaining the first j components
in the MFPCA model.
* PRESS: a vector whose j-th element is the PRESS statistic obtained when retaining the first j
components in the MFPCA model.

References
Capezza C, Lepore A, Menafoglio A, Palumbo B, Vantini S. (2020) Control charts for monitor-
ing ship operating conditions and CO2 emissions based on scalar-on-function regression. Applied

Examples
library(funcharts)
data("air")
air <- lapply(air, function(x) x[1:10, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[fun_covariates], lambda = 1e-2)
y <- rowMeans(air$NO2)
mod <- sof_pc(y, mfdobj_x)
sof_pc_real_time

Get a list of scalar-on-function linear regression models estimated on functional data each evolving up to an intermediate domain point.

Description

This function produces a list of objects, each of them contains the result of applying sof_pc to a scalar response variable and multivariate functional covariates evolved up to an intermediate domain point. See Capezza et al. (2020) for additional details on real-time monitoring.

Usage

sof_pc_real_time(
  y,
  mfd_real_time_list,
  single_min_variance_explained = 0,
  tot_variance_explained = 0.9,
  selection = "PRESS",
  components = NULL,
  ncores = 1
)

Arguments

y                  A numeric vector containing the observations of the scalar response variable.
mfd_real_time_list A list created using get_mfd_df_real_time or get_mfd_list_real_time, denoting a list of functional data objects, each evolving up to an intermediate domain point, with observations of the multivariate functional covariates.
single_min_variance_explained
  See sof_pc.
tot_variance_explained
  See sof_pc.
selection          See sof_pc.
components         See sof_pc.
ncores              If you want parallelization, give the number of cores/threads to be used when creating objects separately for different instants.

Value

A list of lists each produced by sof_pc, corresponding to a given instant.

References

See Also

sof_pc, get_mfd_df_real_time, get_mfd_list_real_time

Examples

library(funcharts)

air <- lapply(air, function(x) x[1:10, , drop = FALSE])
mfdobj_list <- get_mfd_list_real_time(air[c("CO", "temperature")],
                        n_basis = 15,
                        lambda = 1e-2,
                        k_seq = c(0.5, 0.75, 1))

y <- rowMeans(air$NO2)
mod_list <- sof_pc_real_time(y, mfdobj_list)

tensor_product_mfd

Tensor product of two Multivariate Functional Data objects

Description

This function returns the tensor product of two Multivariate Functional Data objects. Each object
must contain only one replication.

Usage

tensor_product_mfd(mfdobj1, mfdobj2 = NULL)

Arguments

mfdobj1 A multivariate functional data object, of class mfd, having only one functional
observation.

mfdobj2 A multivariate functional data object, of class mfd, having only one functional
observation. If NULL, it is set equal to mfdobj1. Default is NULL.

Value

An object of class bifd. If we denote with x(s)=(x_1(s),...x_p(s)) the vector of p functions rep-
resented by mfdobj1 and with y(t)=(y_1(t),...y_q(t)) the vector of q functions represented by
mfdobj2, the output is the vector of pq bivariate functions

f(s,t)=(x_1(s)y_1(t),...x_1(s)y_q(t),...x_p(s)y_1(t),...x_p(s)y_q(t)).
which_ooc

*Get the index of the out of control observations from control charts*

**Description**

This function returns a list for each control chart and returns the id of all observations that are out of control in that control chart.

**Usage**

```r
which_ooc(cclist)
```

**Arguments**

- `cclist` A `data.frame` produced by `control_charts_sof_pc`.

**Value**

A list of as many `data.frame` objects as the control charts in `cclist`. Each data frame has two columns, the `n` contains an index number giving the observation in the phase II data set, i.e. 1 for the first observation, 2 for the second, and so on, while the `id` column contains the id of the observation, which can be general and depends on the specific data set.

**Examples**

```r
library(funcharts)
data("air")
air <- lapply(air, function(x) x[201:300, , drop = FALSE])
fun_covariates <- c("CO", "temperature")
mfdobj_x <- get_mfd_list(air[, fun_covariates],
    n_basis = 15,
    lambda = 1e-2)
y <- rowMeans(air$NO2)
y1 <- y[1:60]
y2 <- y[91:100]
mfdobj_x1 <- mfdobj_x[1:60]
mfdobj_x_tuning <- mfdobj_x[61:90]
mfdobj_x2 <- mfdobj_x[91:100]
mod <- sof_pc(y1, mfdobj_x1)
cclist <- control_charts_sof_pc(mod = mod,
    y_test = y2,
    y_tuning = y2,
    fun_covariates = fun_covariates)
```

**Examples**

```r
library(funcharts)
library(funcharts)

mfdobj1 <- data_sim_mfd(nobs = 1, nvar = 3)
mfdobj2 <- data_sim_mfd(nobs = 1, nvar = 2)
tensor_product_mfd(mfdobj1)
tensor_product_mfd(mfdobj1, mfdobj2)
```

**Examples**

```r
library(funcharts)
mfdobj1 <- data_sim_mfd(nobs = 1, nvar = 3)
mfdobj2 <- data_sim_mfd(nobs = 1, nvar = 2)
tensor_product_mfd(mfdobj1)
tensor_product_mfd(mfdobj1, mfdobj2)
```
mfdobj_x_test = mfdobj_x2, 
mfdobj_x_tuning = mfdobj_x_tuning

which_ooc(cclist)

\[ \text{.mfd} \]

**Extract observations and/or variables from mfd objects.**

**Description**

Extract observations and/or variables from mfd objects.

**Usage**

```r
## S3 method for class 'mfd'
mfdobj[i = TRUE, j = TRUE]
```

**Arguments**

- `mfdobj`: An object of class mfd.
- `i`: Index specifying functional observations to extract or replace. They can be numeric, character, or logical vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer (and hence truncated towards zero). The can also be negative integers, indicating functional observations to leave out of the selection. Logical vectors indicate TRUE for the observations to select. Character vectors will be matched to the argument fdnames[[2]] of mfdobj, i.e. to functional observations’ names.
- `j`: Index specifying functional variables to extract or replace. They can be numeric, logical, or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer (and hence truncated towards zero). The can also be negative integers, indicating functional variables to leave out of the selection. Logical vectors indicate TRUE for the variables to select. Character vectors will be matched to the argument fdnames[[3]] of mfdobj, i.e. to functional variables’ names.

**Details**

This function adapts the fda::"[.fd" function to be more robust and suitable for the mfd class. In fact, whatever the number of observations or variables you want to extract, it always returns a mfd object with a three-dimensional coef array. In other words, it behaves as you would always use the argument drop=FALSE. Moreover, you can extract observations and variables both by index numbers and by names, as you would normally do when using ‘[\' with standard vector/matrices.

**Value**

A mfd object with selected observations and variables.
Examples

```r
library(funcharts)

mfdobj <- data_sim_mfd()
fdobj <- fd(mfdobj$coefs, mfdobj$basis, mfdobj$fdnames)

# The argument 'coef' in 'fd' objects is converted to a matrix when possible.
dim(fdobj[1, 1]$coef)
# Not clear what is the second dimension:
# the number of replications or the number of variables?
dim(fdobj[1, 1:2]$coef)
dim(fdobj[1:2, 1]$coef)

# The argument 'coef' in 'mfd' objects is always a three-dimensional array.
dim(mfdobj[1, 1]$coef)
dim(mfdobj[1, 1:2]$coef)
dim(mfdobj[1:2, 1]$coef)

# Actually, '[.mfd' works as '[.fd' when passing also 'drop = FALSE'
dim(fdobj[1, 1, drop = FALSE]$coef)
dim(fdobj[1, 1:2, drop = FALSE]$coef)
dim(fdobj[1:2, 1, drop = FALSE]$coef)
```
Index

* datasets
  air, 3
  [.mfd, 61

air, 3

cbind.mfd, 4
cont.plot, 13
control_charts_pca, 4, 7, 8, 11, 13, 30, 39, 42, 48
control_charts_pca_mfd_real_time, 7
control_charts_sof_pc, 9, 11–13, 30, 39, 42, 60
control_charts_sof_pc_real_time, 11, 40, 41
data_sim_mfd, 14

fd, 33, 34
fof_pc, 15, 17, 18
fof_pc_real_time, 17, 49, 50
funcharts, 19

geom_line, 20
geom_mfd, 19, 42
get_mfd_array, 20, 22, 28
get_mfd_array_real_time, 21
get_mfd_df, 21, 23, 25, 26, 29
get_mfd_df_real_time, 8, 12, 18, 25, 37, 49, 50, 58, 59
get_mfd_fd, 26
get_mfd_list, 21, 24, 27, 28, 29
get_mfd_list_real_time, 18, 28, 59
get_ooc, 30
get_sof_pc_outliers, 31
ggplot, 20

inprod_mfd, 31
inprod_mfd_diag, 32
is.mfd, 33

mean.fd, 53
mfd, 28, 33

norm.mfd, 35

pca.fd, 36
pca_mfd, 8, 36, 37
pca_mfd_real_time, 7, 8, 37
plot_bifd, 38
plot_bootstrap_sof_pc, 38
plot.control_charts, 39
plot_control_charts_real_time, 40
plot_mfd, 41
plot_mon, 42
plot_pca.mfd, 43
predict_fof_pc, 44
predict_sof_pc, 45

rbind.mfd, 46
regr_cc_fof, 6, 13, 30, 39, 42, 47, 49, 50
regr_cc_fof_real_time, 40, 41, 49
regr_cc_sof, 9, 11, 13, 30, 39, 42, 51

scale, 53
scale.mfd, 36, 52
sd.fd, 53
sim.funcharts, 55
simulate_mfd, 53, 55
sof_pc, 39, 56, 58, 59
sof_pc_real_time, 12, 13, 58
tensor_product_mfd, 59

which_ooc, 60

63