Package 'ehymet'

Title Methodologies for Functional Data Based on the Epigraph and
Hypograph Indices

Version 0.1.1

Description Implements methods for functional data analysis based on the epigraph
and hypograph indices. These methods transform
functional datasets, whether in one or multiple dimensions, into multivariate
datasets. The transformation involves applying the epigraph, hypograph, and
their modified versions to both the original curves and their first and second
derivatives. The calculation of these indices is tailored to the dimensionality
of the functional dataset, with special considerations for dependencies between
dimensions in multidimensional cases. This approach extends traditional multivariate
data analysis techniques to the functional data setting. A key application of
this package is the EHyClus method, which enhances clustering analysis for
functional data across one or multiple dimensions using the epigraph and
hypograph indices. See Pulido et al. (2023) <doi:10.1007/s11222-023-10213-7>
and Pulido et al. (2024) <doi:10.48550/arXiv.2307.16720>.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/bpulidob/ehymet,

https://bpulidob.github.io/ehymet/

BugReports https://github.com/bpulidob/ehymet/issues

Depends R (>= 4.1)

Imports clusterCrit, kernlab, stats, tf

Suggests ggplot2, knitr, MASS, parallel, rmarkdown, testthat (>=
3.0.0), tidyr

RoxygenNote 7.3.2

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Belen Pulido [aut, cre] (<https://orcid.org/0000-0003-2105-959X>),
Jose Ignacio Diez [ctr]

Maintainer Belen Pulido <bpulidob4@gmail.com>

Repository CRAN

Date/Publication 2024-11-26 23:00:13 UTC

Contents

---

clustering_validation	*Create a table containing four validation metrics for clustering: Purity, F-measure and Rand Index (RI) and Adjusted Rand Index (ARI). This function considers pairs of points*

Description

Create a table containing four validation metrics for clustering: Purity, F-measure and Rand Index (RI) and Adjusted Rand Index (ARI). This function considers pairs of points

Usage

```
clustering_validation(clusters, true_labels, digits = 4)
```

Arguments

clusters	The clusters predicted by the clustering method.
true_labels	Atomic vector with the true labels of the data.
digits	Number of digits for rounding.

Value

A list containing values for Purity, F-measure, RI and ARI.

Examples

```
set.seed(1221)
vars <- list(c("dtaEI", "dtaMEI"))
data <- sim_model_ex1()
true_labels <- c(rep(1, 50), rep(2, 50))
data_ind <- generate_indices(data)
clus_kmeans <- clustInd_kmeans(data_ind, vars)
cluskmeans_mahalanobis_dtaEIdtaMEI <- clus_kmeans$kmeans_mahalanobis_dtaEIdtaMEI$cluster
clustering_validation(cluskmeans_mahalanobis_dtaEIdtaMEI, true_labels)
```

---

clustInd_hierarch	*Perform hierarchical clustering for a different combinations of indices, method and distance*

Description

Perform hierarchical clustering for a different combinations of indices, method and distance

Usage

```
clustInd_hierarch(
  ind_data,
  vars_combinations,
  method_list = c("single", "complete", "average", "centroid", "ward.D2"),
  dist_vector = c("euclidean", "manhattan"),
  n_cluster = 2,
  true_labels = NULL,
  n_cores = 1
)
```

Arguments

ind_data   Dataframe containing indices applied to the original data and its first and second derivatives. See generate_indices.

vars_combinations

     list containing one or more combinations of indices in ind_data. If it is non-named, the names of the variables are set to vars1, ..., varsk, where k is the number of elements in vars_combinations.

method_list   list of clustering methods.

dist_vector   list of distance metrics.

n_cluster   number of clusters to generate.

true_labels   Vector of true labels for validation (if it is not known true_labels is set to NULL)

n_cores   Number of cores to do parallel computation. 1 by default, which mean no parallel execution.

Value

A `list` containing hierarchical clustering results for each configuration.

Examples

```
vars1 <- c("dtaEI", "dtaMEI")
vars2 <- c("dtaHI", "dtaMHI")
data <- ehymet::sim_model_ex1()
data_ind <- generate_indices(data)
clustInd_hierarch(data_ind, list(vars1, vars2))
```

---

clustInd_kkmeans   *Kernel k-means clustering using indices*

Description

Perform kernel kmeans clustering for a different combinations of indices and kernel

Usage

```
clustInd_kkmeans(
  ind_data,
  vars_combinations,
  kernel_list = c("rbfdot", "polydot"),
  n_cluster = 2,
  true_labels = NULL,
  n_cores = 1
)
```

Arguments

ind_data	Dataframe containing indices applied to the original data and its first and second derivatives. See generate_indices.
vars_combinations	
	`list` containing one or more combinations of indices in `ind_data`. If it is non-named, the names of the variables are set to vars1, ..., varsk, where k is the number of elements in `vars_combinations`.
kernel_list	List of kernels
n_cluster	Number of clusters to create
true_labels	Vector of true labels for validation (if it is not known true_labels is set to NULL)
n_cores	Number of cores to do parallel computation. 1 by default, which mean no parallel execution.

Value

A `list` containing kernel-kmeans clustering results for each configuration.

Examples

```
vars1 <- c("dtaEI", "dtaMEI")
vars2 <- c("dtaHI", "dtaMHI")
data <- ehymet::sim_model_ex1()
data_ind <- generate_indices(data)
clustInd_kkmeans(data_ind, list(vars1, vars2))
```

---

clustInd_kmeans *K-means clustering with indices*

Description

Perform k-means clustering for a different combinations of indices and distances.

Usage

```
clustInd_kmeans(
  ind_data,
  vars_combinations,
  dist_vector = c("euclidean", "mahalanobis"),
  n_cluster = 2,
  init = "random",
  true_labels = NULL,
  n_cores = 1
)
```

Arguments

ind_data   Dataframe containing indices applied to the original data and its first and second
                    derivatives. See [generate_indices](#).

vars_combinations

                    list containing one or more combinations of indices in ind_data. If it is non-
                    named, the names of the variables are set to vars1, ..., varsk, where k is the
                    number of elements in vars_combinations.

dist_vector   Atomic vector of distance metrics. The possible values are, "euclidean", "maha-
                    lanobis" or both.

n_cluster   Number of clusters to create.

init   Centroids initialization meathod. It can be "random" or "kmeanspp".

true_labels   Vector of true labels for validation. (if it is not known true_labels is set to NULL)

n_cores   Number of cores to do parallel computation. 1 by default, which mean no par-
                    allel execution.

Value

A list containing hierarchical clustering results for each configuration

A list containing kmeans clustering results for each configuration

Examples

```
vars1 <- c("dtaEI", "dtaMEI")
vars2 <- c("dtaHI", "dtaMHI")
data <- ehymet::sim_model_ex1()
data_ind <- generate_indices(data)
clustInd_kmeans(data_ind, list(vars1, vars2))
```

---

clustInd_spc   *Spectral clustering using indices*

Description

Perform spectral clustering for a different combinations of indices and kernels

Usage

```
clustInd_spc(
  ind_data,
  vars_combinations,
  kernel_list = c("rbfdot", "polydot"),
  n_cluster = 2,
  true_labels = NULL,
  n_cores = 1
)
```

Arguments

ind_data   Dataframe containing indices applied to the original data and its first and second
                  derivatives. See generate_indices.

vars_combinations

                  list containing one or more combinations of indices in ind_data. If it is non-
                  named, the names of the variables are set to vars1, ..., varsk, where k is the
                  number of elements in vars_combinations.

kernel_list   List of kernels

n_cluster   Number of clusters to create

true_labels   Vector of true labels for validation (if it is not known true_labels is set to NULL)

n_cores   Number of cores to do parallel computation. 1 by default, which mean no par-
                  allel execution.

Value

A list containing kkmeans clustering results for each configuration

Examples

```
vars1 <- c("dtaEI", "dtaMEI")
vars2 <- c("dtaHI", "dtaMHI")
data <- ehymet::sim_model_ex1()
data_ind <- generate_indices(data)
clustInd_spc(data_ind, list(vars1, vars2))
```

EHyClus   *Clustering using Epigraph and Hypograph indices*

Description

It creates a multivariate dataset containing the epigraph, hypograph and/or its modified versions on the curves and derivatives and then perform hierarchical clustering, kmeans, kernel kmeans, and spectral clustering

Usage

```
EHyClus(
  curves,
  vars_combinations,
  k = 30,
  n_clusters = 2,
  bs = "cr",
  clustering_methods = c("hierarch", "kmeans", "kkmeans", "spc"),
  l_method_hierarch = c("single", "complete", "average", "centroid", "ward.D2"),
  l_dist_hierarch = c("euclidean", "manhattan"),
  l_dist_kmeans = c("euclidean", "mahalanobis"),
  l_kernel = c("rbfdot", "polydot"),
  true_labels = NULL,
  only_best = FALSE,
  verbose = FALSE,
  n_cores = 1,
  ...
)
```

Arguments

curves   Dataset containing the curves to apply a clustering algorithm. The functional dataset can be one dimensional ($n \times p$) where n is the number of curves and p the number of time points, or multidimensional ($n \times p \times q$) where q represents the number of dimensions in the data

vars_combinations

If list, each element of the list should be an atomic vector of strings with the names of the variables. Combinations with non-valid variable names will be discarded. If the list is non-named, the names of the variables are set to vars1,

	..., varsk, where k is the number of elements in `vars_combinations`. If "auto" is provided, an unique combination of variable will be found. If not provided, generic combinations of variables will be used. They will not be the same for uni-dimensional and multi-dimensional problems.
k	Number of basis functions for the B-splines. If equals to `0`, the number of basis functions will be automatically selected.
n_clusters	Number of clusters to generate.
bs	A two letter character string indicating the (penalized) smoothing basis to use. See `smooth.terms`.
clustering_methods	
	character vector specifying at least one of the following clustering methods to be computed: "hierarch", "kmeans", "kkmeans" or "spc".
l_method_hierarch	
	`list` of clustering methods for hierarchical clustering.
l_dist_hierarch	
	`list` of distances for hierarchical clustering.
l_dist_kmeans	`list` of distances for kmeans clustering.
l_kernel	`list` of kernels for kkmeans or spc.
true_labels	Numeric vector of true labels for validation. If provided, evaluation metrics are computed in the final result.
only_best	`logical` value. If `TRUE` and `true_labels` is provided, the function will return only the result for the best clustering method based on the Rand Index. Defaults to `FALSE`.
verbose	If `TRUE`, the function will print logs for about the execution of some clustering methods. Defaults to `FALSE`.
n_cores	Number of cores to do parallel computation. 1 by default, which mean no parallel execution. Must be an integer number greater than 1.
...	Additional arguments for tfb. See `tfb`.

Value

A `list` containing the clustering partition for each method and indices combination and, if `true_labels` is provided a data frame containing the time elapsed for obtaining a clustering partition of the indices dataset for each methodology. Also, the number of generated clusters and the combinations of variables used can be seen as attributes of this object.

Examples

```
# univarariate data without labels
curves <- sim_model_ex1(n = 10)
vars_combinations <- list(c("dtaEI", "dtaMEI"), c("dtaHI", "dtaMHI"))
EHyClus(curves, vars_combinations = vars_combinations)

# multivariate data with labels
curves <- sim_model_ex2(n = 5)
true_labels <- c(rep(1, 5), rep(2, 5))
```

```
vars_combinations <- list(c("dtaMEI", "ddtaMEI"), c("dtaMEI", "d2dtaMEI"))
res <- EHyClus(curves, vars_combinations = vars_combinations, true_labels = true_labels)
res$cluster # clustering results

# multivariate data and generic (default) vars_combinations
curves <- sim_model_ex2(n = 5)
EHyClus(curves)
```

EI   *Epigraph Index (EI) for a functional dataset*

Description

The Epigraph Index of a curve x is one minus the proportion of curves in the sample that are above x.

Usage

```
EI(curves, ...)
```

Arguments

curves   matrix where each row represents a curve, and each column represents values
             along the curve or array with dimension $n \times p \times q$ with n curves, p values along
             the curve, and q dimensions.

...   Ignored.

Value

numeric vector containing the EI for each curve.

Examples

```
x <- matrix(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7), ncol = 3, nrow = 4)
EI(x)

y <- array(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7, -1, -5, -6, 2, 3, 0, -1, 0, 2, -1, -2, 0),
  dim = c(3, 4, 2)
)
EI(y)
```

---

generate_indices   *Create a dataset with indices from a functional dataset in one or multiple dimensions*

Description

Create a dataset with indices from a functional dataset in one or multiple dimensions

Usage

```
generate_indices(
  curves,
  k,
  bs = "cr",
  indices = c("EI", "HI", "MEI", "MHI"),
  n_cores = 1,
  ...
)
```

Arguments

curves	matrix with dimension $n \times p$ in the case of a one-dimensional functional dataset, or array of dimension $n \times p \times q$ in the case of a multivariate functional dataset. n represents the number of curves, p the number of values along the curve, and in the second case, q is the number of dimensions.
k	Number of basis functions for the B-splines. If equals to 0, the number of basis functions will be automatically selected.
bs	A two letter character string indicating the (penalized) smoothing basis to use. See smooth.terms.
indices	Set of indices to be applied to the dataset. They should be any between EI, HI, MEI and MHI.
n_cores	Number of cores to do parallel computation. 1 by default, which mean no parallel execution. Must be an integer number greater than 1.
...	Additional arguments for tfb. See tfb.

Value

A dataframe containing the indices provided in indices for original data, first and second derivatives

Examples

```
# 3-dimensional array
x1 <- array(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7, -1, -5, -6, 2, 3, 0, -1, 0, 2, -1, -2, 0),
  dim = c(3, 4, 2)
)
```

```
generate_indices(x1, k = 4)

# matrix
x2 <- matrix(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7), nrow = 3, ncol = 4)
generate_indices(x2, k = 4)

# using additional parameter for tf::tfb
curves <- sim_model_ex1(n = 10)
generate_indices(
  curves = curves,
  k = 20,
  bs = "bs",
  m = c(3,2),        # additional parameter for tfb
  penalized = FALSE  # additional parameter for tfb
)
```

HI   *Hypograph Index (HI) for a functional dataset*

Description

The Hypograph Index of a curve x is the proportion of curves in the sample that are below x.

Usage

```
HI(curves, ...)
```

Arguments

curves   matrix where each row represents a curve, and each column represents values
                along the curve or array with dimension $n \times p \times q$ with n curves, p values along
                the curve, and q dimensions.
...   Ignored.

Value

numeric vector containing the HI for each curve.

Examples

```
x <- matrix(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7), ncol = 3, nrow = 4)
HI(x)

y <- array(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7, -1, -5, -6, 2, 3, 0, -1, 0, 2, -1, -2, 0),
  dim = c(3, 4, 2)
)
HI(y)
```

MEI   *Modified Epigraph Index (MEI) for functional dataset.*

Description

The Modified Epigraph Index of a curve x is one minus the proportion of "time" the curves in the sample are above x.

Usage

```
MEI(curves, ...)
```

Arguments

curves   matrix where each row represents a curve, and each column represents values along the curve or an array with dimension $n \times p \times q$ with n curves, p values along the curve, and q dimensions.

...   Ignored.

Value

numeric vector containing the MEI for each curve.

Examples

```
x <- matrix(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7), ncol = 3, nrow = 4)
MEI(x)
y <- array(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7, -1, -5, -6, 2, 3, 0, -1, 0, 2, -1, -2, 0),
  dim = c(3, 4, 2)
)
MEI(y)
```

MHI   *Modified Hypograph Index (MHI) for a functional dataset*

Description

The Modified Hypograph Index of a curve x is the proportion of "time" the curves in the sample are below x.

Usage

```
MHI(curves, ...)
```

Arguments

curves   `matrix` where each row represents a curve, and each column represents values along the curve or an `array` with dimension $n \times p \times q$ with n curves, p values along the curve, and q dimensions.

...   Ignored.

Value

`numeric` `vector` containing the MHI for each curve.

Examples

```
x <- matrix(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7), ncol = 3, nrow = 4)
MHI(x)
y <- array(c(1, 2, 3, 3, 2, 1, 5, 2, 3, 9, 8, 7, -1, -5, -6, 2, 3, 0, -1, 0, 2, -1, -2, 0),
  dim = c(3, 4, 2)
)
MHI(y)
```

---

sim_model_ex1   *Function for generating functional data in one dimension*

Description

Each dataset has 2 groups with n curves each, defined in the interval $t = [0, 1]$ with p equidistant points. The first n curves are generated fron the following model $X_1(t) = E_1(t) + e(t)$ where $E_1(t) = E_1(X(t)) = 30t^{\frac{3}{2}}(1 - t)$ is the mean function and $e(t)$ is a centered Gaussian process with covariance matrix $Cov(e(t_i), e(t_j)) = 0.3 \exp(-\frac{|t_i - t_j|}{0.3})$ The remaining 50 functions are generated from model i_sim with i_sim $\in \{1, \ldots, 8\}$. The first three models contain changes in the mean, while the covariance matrix does not change. Model 4 and 5 are obtained by multiplying the covariance matrix by a constant. Model 6 is obtained from adding to $E_1(t)$ a centered Gaussian process $h(t)$ whose covariance matrix is given by $Cov(e(t_i), e(t_j)) = 0.5 \exp(-\frac{|t_i - t_j|}{0.2})$. Model 7 and 8 are obtained by a different mean function.

Model 1. $X_1(t) = 30t^{\frac{3}{2}}(1 - t) + 0.5 + e(t)$.

Model 2. $X_2(t) = 30t^{\frac{3}{2}}(1 - t) + 0.75 + e(t)$.

Model 3. $X_3(t) = 30t^{\frac{3}{2}}(1 - t) + 1 + e(t)$.

Model 4. $X_4(t) = 30t^{\frac{3}{2}}(1 - t) + 2e(t)$.

Model 5. $X_5(t) = 30t^{\frac{3}{2}}(1 - t) + 0.25e(t)$.

Model 6. $X_6(t) = 30t^{\frac{3}{2}}(1 - t) + h(t)$.

Model 7. $X_7(t) = 30t(1 - t)^2 + h(t)$.

Model 8. $X_8(t) = 30t(1 - t)^2 + e(t)$.

Usage

```
sim_model_ex1(n = 50, p = 30, i_sim = 1)
```

Arguments

n	Number of curves to generate for each of the two groups. Set to 50 by default.
p	Number of grid points of the curves. Curves are generated over the interval $[0, 1]$. Set to 30 grid point by default.
i_sim	Integer set to $1, \dots, 8$.

Value

data matrix of size $2n \times p$.

Examples

```
sm1 <- sim_model_ex1()
dim(sm1)
```

---

sim_model_ex2   *Function for generating functional data in one or multiple dimension*

Description

The function can generate one-dimensional or multi-dimensional curves. For `i_sim` 1 or 2, one-dimensional curves are generated. For `i_sim` 3 or 4, multi-dimensional curves are generated.

Usage

```
sim_model_ex2(n = 50, p = 150, i_sim = 1)
```

Arguments

n	Number of curves to generate for each of the two groups. Set to 50 by default.
p	Number of grid points of the curves. Curves are generated over the interval $[0, 1]$. Set to 150 grid point by default.
i_sim	Integer set to $1, \dots, 4$ NULL by default in which case a seed is not set.

Value

data matrix of size $2n \times p$ if $i\_sim \in 1, 2$ or an array of dimensions $2n \times p \times 2$ if $i\_sim \in 3, 4$.

Examples

```
sm1 <- sim_model_ex2()
dim(sm1) # This should output (100, 150) by default, since n = 50 and p = 150

sm4 <- sim_model_ex2(i_sim = 4)
dim(sm4) # This should output (100, 150, 2) by default, since n = 50 and p = 150
```

Index