Package 'SparseFunClust'

January 20, 2025

Title Sparse Functional Clustering

Version 1.0.0

Description Provides a general framework for performing sparse functional clustering as originally described in Floriello and Vitelli (2017) <doi:10.1016/j.jmva.2016.10.008>, with the possibility of jointly handling data misalignment (see Vitelli, 2019, <doi:10.48550/arXiv.1912.00687>).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Imports cluster

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Valeria Vitelli [aut],
Waldir Leoncio [cre]

Maintainer Waldir Leoncio <w.l.netto@medisin.uio.no>

Repository CRAN

Date/Publication 2023-03-28 19:30:02 UTC

Contents

cer *CER function*

Description

Given two partitions P and Q, `cer(P, Q)` measures how well they agree, the lower the better. It is rigorously defined as the proportion of pairwise disagreements in the two partitions (i.e., how many, out of all the possible couples of elements in the sample, are localized in the same cluster in one partition and in a different one in the other partition).

Usage

```
cer(P, Q)
```

Arguments

P	first vector of cluster assignments (length n)
Q	second vector of cluster assignments (length n)

Value

The CER index, which is a number between 0 and 1, and also equal to 1 - Rand index (Rand, 1971), a popular measure of the goodness of a clustering.

References

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association, 66(336), 846-850.

Examples

```
set.seed(8988327)
x <- seq(0, 1, len = 500)
out <- generate.data.FV17(50, x)
result <- SparseFunClust(out$data, x, K = 2, do.alignment = FALSE)
cer(out$true.partition, result$labels)
```

generate.data.FV17 *Data generation: no-misalignment case*

Description

this function generates a set of simulated functional data in 2 clusters that reproduce the examples in Simulations 2A and 2B in Floriello & Vitelli (2017).

Usage

```
generate.data.FV17(n, x, paramC = 0.5, plots = FALSE)
```

Arguments

n	number of curves
x	curves' domain
paramC	proportion of cluster overlap (default 0.5, as in Simulation 2A)
plots	boolean; should plots be drawn (FALSE default)

Value

a list including:

- $data matrix (n x length(x)) with the simulated data
- $true.partition vector (length = n) with the true cluster assignments

Examples

```
generate.data.FV17(5, seq(0, 1, len = 3))
```

SparseFunClust *Compute Sparse Functional Clustering & Alignment*

Description

Compute Sparse Functional Clustering & Alignment

Usage

```
SparseFunClust(
  data,
  x,
  K,
  do.alignment,
  funct.measure = "L2",
  clust.method = "kmea",
  m.prop = 0.3,
  tuning.m = FALSE,
  tuning.par = list(mbound = NULL, nperm = 20),
  perc = 0.03,
  tol = 0.01,
  template.est = "raw",
  n.out = 500,
  iter.max = 50,
  vignette = TRUE
)
```

Arguments

`data`	matrix representing the functions (n x p)
`x`	matrix giving the domain of each function (n x p), or a p-dimensional vector giving the common domain
`K`	number of clusters
`do.alignment`	boolean (should alignment be performed?)
`funct.measure`	the functional measure to be used to compare the functions in both the clustering and alignment procedures; can be 'L2' or 'H1' (default 'L2'); see Vitelli (2019) for details
`clust.method`	the clustering method to be used; can be: 'kmea' for k-means clustering,'pam','hier' for hierarchical clustering
`m.prop`	the sparsity parameter (proportion of unrelevant domain where w(x) = 0); default 30%
`tuning.m`	boolean (should the sparsity parameter be tuned via a permutation-based approach?)
`tuning.par`	list of settings for the tuning of the sparsity parameter (defaults to `list(mbound = NULL, nperm = 20)`: mbound = max value of the sparsity parameter to be tested, default 60%; nperm = number of permutations to be performed in the tuning, default 20
`perc`	alignment parameter (max proportion of shift / dilation at each iter of the warping procedure) –> (default 3%)
`tol`	tolerance criterion on the weighting function to exit the loop (default 1%)
`template.est`	text string giving choices for the template estimation method
`n.out`	number of abscissa points on which w(x) is estimated (default 500)
`iter.max`	maximum number of iterations of the clustering loop (default 50)
`vignette`	boolean (should the algorithm progress be reported?)

Value

A list, with elements:

template matrix (dim=K x n.out) with the final cluster templates

temp.abscissa vector (length=n.out) of the abscissa values on which the template is defined

labels vector (length=n) of the cluster assignments

warping matrix (dim=n x 2) with the intercept (1st column) and slope (2nd column) of the estimated warping function for each of the n curves

reg.abscissa matrix (dim=n x n.out) of each of the n curves registered abscissa

distance vector (length=n) of each curve's final distance to the assigned cluster template

w vector (length=n.out) of the estimated weighting function w(x)

x.bcss vector (length=n.out) of the final point-wise between-cluster sum-of-squares

Note

`data`:

1. assumed to be a vectorized version of the functional data AFTER smoothing
2. when using the H1 functional measure, assumed to include the functions FIRST DERIVATIVES
3. when using the H1 functional measure, it supports multidimensional functions R -> R^d, then data can be an array (n x p x d)]

`funct.measure`: 'H1' only supported with alignment

`clust.method`: 'pam' and 'hier' only supported for the case of NO ALIGNMENT

`m.prop`: needs to be a proportion for compatibility with alignment, values > 1 not supported

`tuning.m`: tuning only supported for the case of NO ALIGNMENT

`tuning.par`:

- `mbound` must be lower than 1; the minimal value tested is 0
- `nperm` > 50 is unadvisable for computational reasons

`perc`: 5% is already extreme; don't set this above 8-10%

`template.est`:

1. only supported with H1 measure + ALIGNMENT
2. currently 2 choices are supported:'raw' or 'loess'. 'raw' just computes the vector means across functions (default choice); 'loess' estimates the template via the R loess function

Examples

```
set.seed(8988327)
x <- seq(0, 1, len = 500)
out <- generate.data.FV17(50, x)
result <- SparseFunClust(out$data, x, K = 2, do.alignment = FALSE)
str(result)
```

Index