Package ‘fdWasserstein’

February 6, 2024

Type Package

Title Application of Optimal Transport to Functional Data Analysis

Version 1.0

Date 2024-01-08

Author Valentina Masarotto [aut, cph, cre],
Guido Masarotto [aut, cph]

Maintainer Valentina Masarotto <v.masarotto@math.leidenuniv.nl>

Suggests future

Depends R (>= 3.5.0)

Description
These functions were developed to support statistical analysis on functional covariance operators.
The package contains functions to:
- compute 2-Wasserstein distances between Gaussian Processes as in Masarotto, Panaretos & Zemel (2019) <doi:10.1007/s13171-018-0130-1>;
- compute the Wasserstein barycenter (Frechet mean) as in Masarotto, Panaretos & Zemel (2019) <doi:10.1007/s13171-018-0130-1>;
- perform analysis of variance testing procedures for functional covariances and tangent space principal component analysis of covariance operators as in Masarotto, Panaretos & Zemel (2022) <arXiv:2212.04797>.
- perform a soft-clustering based on the Wasserstein distance where functional data are classified based on their covariance structure as in Masarotto & Masarotto (2023) <doi:10.1111/sjos.12692>.

NeedsCompilation no

Repository CRAN

Date/Publication 2024-02-06 17:00:02 UTC

R topics documented:

`fdWasserstein-package` *Application of Optimal Transport to Functional Data Analysis*

Description

A package containing functions developed to support statistical analysis on functional covariance operators. In particular,

- Function `dwasserstein` computes the Wasserstein-Procrustes distance between two covariances.
- Function `gaussBary` computes the Frechet mean of K covariances with respect to the Procrustes metrics (equivalently, the Wasserstein barycenter of centered Gaussian processes with corresponding covariances) via steepest gradient descent. See Masarotto, Panaretos & Zemel (2019).
- Function `tangentPCA` performs the tangent space principal component analysis considered in Masarotto, Panaretos & Zemel (2022).
- Function `wassersteinTest` lets to test the null hypothesis that K covariances are equal using the methodology suggested by Masarotto, Panaretos & Zemel (2022).
- Function `wassersteinCluster` implements the soft partion procedure proposed by Masarotto & Masarotto (2023).

Author(s)

Valentina Masarotto [aut, cph, cre], Guido Masarotto [aut, cph]

Maintainer: Valentina Masarotto <v.masarotto@math.leidenuniv.nl>

References

Masarotto, V., Panaretos, V.M. & Zemel, Y. (2019) "Procrustes Metrics on Covariance Operators and Optimal Transportation of Gaussian Processes", *Sankhya A* **81**, 172-213 doi:10.1007/s13171-01801301

Masarotto, V., Panaretos, V.M. & Zemel, Y. (2022) "Transportation-Based Functional ANOVA and PCA for Covariance Operators", *arXiv*, `https://arxiv.org/abs/2212.04797`

Masarotto, V. & Masarotto, G. (2023) "Covariance-based soft clustering of functional data based on the Wasserstein-Procrustes metric", *Scandinavian Journal of Statistics*, doi:10.1111/sjos.12692.

`dwasserstein` *2-Wasserstein distance*

Description

Computes the 2-Wasserstein distance between the (covariance) matrices A and B.

Usage

```
dwasserstein(A, B)
```

Arguments

`A,B`	Two symmetric positive semi-definite matrices.

Value

A numeric object with the 2-Wasserstein distance of A and B.

Author(s)

Valentina Masarotto, Guido Masarotto

References

Masarotto, V., Panaretos, V.M. & Zemel, Y. (2019) "Procrustes Metrics on Covariance Operators and Optimal Transportation of Gaussian Processes", *Sankhya A* **81**, 172-213 doi:10.1007/s13171-01801301

See Also

`gaussBary`

Examples

```
n <- 10
matrices <- rWishart(2,n,diag(n))
A <- matrices[,,2]
B <- matrices[,,1]
dwasserstein(A,B)
dwasserstein(A, 10*crossprod(B))
```

gaussBary — *Wasserstein barycenter between Gaussian Processes*

Description

Computes the Frechet mean between covariance operators with respect to the Procrustes metrics (equivalently, a Wasserstein barycenter of centered Gaussian processes with corresponding covariances) via steepest gradient descent.

Usage

```
gaussBary(sigma, w = rep(1, dim(sigma)[3]), gamma, sigma0.5,
  max.iter = 30, eps = 1e-08, silent = max.iter == 0)
```

Arguments

`sigma`	An MxMxK array containing the K covariances.
`w`	Optional. A vector of weights of length K. If missing, each matrix is given equal weight 1.
`gamma`	Optional. Initialisation point for the gradient descent algorithm.
`sigma0.5`	Optional. An array containing the square roots of the matrices in sigma if available. The square roots are computed by `gaussBary` if `sigma0.5` is missing.
`max.iter`	Maximum number of gradient descent iterations.
`eps`	Iterations stop when the relative decrease of the objective function in two consecutive iterations is less than ‘eps‘.
`silent`	If `FALSE` returns a warning if maximal number of iteration is reached.

Value

A list of 2 containing:

`gamma`	The MxM Frechet mean.
`iter`	Number of iterations needed to reach convergence, numeric.

Note

We thank Yoav Zemel for the first version of the code.

Author(s)

Valentina Masarotto, Guido Masarotto

References

Masarotto, V., Panaretos, V.M. & Zemel, Y. (2019) "Procrustes Metrics on Covariance Operators and Optimal Transportation of Gaussian Processes", *Sankhya A* **81**, 172-213 doi:10.1007/s13171-01801301

Examples

```
M <- 5
K <- 4

sigma <- rWishart(M, df = K, Sigma = diag(K))

gaussBary(sigma)
```

Phoneme *Phoneme data*

Description

The dataset comprises 4509 log-periodograms computed from digitalized speech frames. Each log-periodograms is of length 256, and is based on the pronunciation of one of the following five phonemes: "sh", "dcl", "iy", "aa" and "ao".

Usage

```
data(phoneme)
```

Format

- `logPeriodogram`: a 4509x256 matrix containing the log-periodograms.
- `Phoneme`: a vector of length 4509 containing the phonemes.

Source

The data set was downloaded from the "*Elements of statistical learning*" website at https://hastie.su.domains/ElemStatLearn/

References

T. Hastie and R. Tibshirani and J. Friedman (2009) *The elements of statistical learning: Data mining, inference and prediction*, 2nd edn, New York: Springer.

Examples

```
data(phoneme)
old <- par(mfrow=c(3,2))
for (i in unique(Phoneme))
    matplot(t(logPeriodogram[Phoneme==i,]), type="l",
            xlab="", ylab="", ylim=c(0,30), main=i)
par(old)
```

tangentPCA *Tangent space principal component analysis*

Description

The function performs a standard PCA of K covariances after projecting them on the tangent space at their Wasserstein barycenter. Rationale and details are given in Masarotto, Panaretos & Zemel (2019, 2022)

Usage

```
tangentPCA(sigma, max.iter=30)
```

Arguments

sigma	An MxMxK array containing the K covariances.
max.iter	Maximum number of gradient descent iterations used to compute the Wasserstein barycenter of the covariances in sigma.

Value

A standard prcomp object with added a MxMxK array containing the eigenvectors projected back to the covariances space.

Author(s)

Valentina Masarotto

References

Masarotto, V., Panaretos, V.M. & Zemel, Y. (2022) "Transportation-Based Functional ANOVA and PCA for Covariance Operators", *arXiv*, https://arxiv.org/abs/2212.04797

See Also

gaussBary, prcomp

Examples

```
# Example taken from https://arxiv.org/abs/2212.04797  .
data(phoneme)
# resampling the log-periodograms
# 12 sample covariances for each phoneme
# each estimated on 50 curves
set.seed(12345)
nsubsamples <- 12
n <- 50
```

```
gg <- unique(Phoneme)
nphonemes <- length(gg)
K <- n*nsubsamples*nphonemes
M <- NCOL(logPeriodogram)
Sigma <- array(dim=c(M, M, nphonemes*nsubsamples))
r <- 0
for (l in gg) {
  for (i in 1:nsubsamples) {
      r <- r+1
      Sigma[,,r] <- cov(logPeriodogram[sample(which(Phoneme==l),n), ])
  }
}
pca <- tangentPCA(Sigma, max.iter=3)
summary(pca)
plot(pca)
# See https://arxiv.org/abs/2212.04797 for the interpretation
# of the figure
pairs(pca$x[,1:5], col=rep(1:nphonemes, rep(nsubsamples, nphonemes)))
```

`wassersteinCluster` *Soft clustering of covariance operators.*

Description

Computes the soft cluster solutions for different values of the number of clusters K.

Usage

```
wassersteinCluster(data, grp,
                    kmin = 2, kmax = 10,
                    E = -0.75 * (0.95 * log(0.95) +
                         0.05 * log(0.05)) + 0.25 * log(2),
                    nstart = 5, nrefine = 5, ntry = 0,
                    max.iter = 20, tol = 0.001,
                    nreduced = length(unique(grp)),
                    nperm = 0,
                    add.sigma = FALSE,
                    use.future = FALSE, verbose = TRUE)

trimmedAverageSilhouette(a, plot = TRUE)
```

Arguments

`data`	A N times M matrix containing the N sample curves; M denotes the number of points of the grid on which the curves are available.
`grp`	A vector or factor of length N; a covariance operator is estimated for each level of grp.

- `kmin,kmax` A pair of integer defining the desired number of clusters. A solution is computed for K=kmin,...,kmax.
- `E` The desired average entropy.
- `nstart, nrefine, ntry` The integers used during the initialization search. If ntry=0, then 'ntry' is set to 'round(1+N/K)'.
- `max.iter` Maximum number of block descend iterations.
- `tol` Iterations stop when the relative decrease of the objective function in two consecutive iterations is less than 'tol'.
- `nreduced` The number of covariances used to estimate the cluster barycenters.
- `nperm` The number of permutation used to approximate the reference distribution of max TASW.
- `add.sigma` Should the sample covariances be returned?
- `use.future` Use or not use package 'future' to parallelize the computation? See note.
- `verbose` If 'verbose==TRUE', information on the progress of the optimization are shown.
- `a` A list returned by 'wassersteinCluster'.
- `plot` If 'plot==TRUE', the TASW profile is plotted.

Details

See Masarotto & Masarotto (2023) for the algorithm details.

Value

'wassersteinCluster' returns a list of length kmax-kmin+1. The ith element is a list describing the cluster solution obtained for k=kmin+i-1, and containing:

- `K,E,eta` the number of cluster, the average entropy and the corresponding value of 'eta';
- `w` the N times K soft partition matrix;
- `g` a M times M times K array with the cluster barycenters;
- `d` a N times K matrices containing the distances between the N sample covariances and the K cluster barycenters;
- `obj` 'obj': the minimum value of the objective function.

The list may have the following attributes:

- `df` the degree of freedom of the sample operators (a vector). Always present.
- `sample.covariances` a list contaning the sample operators (as a 3-dimensional array); only present if add.sigma=TRUE;
- `tasw.test` a list containing the value of maxTASW computed from the data (a scalar), the nperm values of of maxTASW obtained by permutation (a vector), and the corresponding p-value (a scalar); only present if nperm>0.

'trimmedAverageSilhouette' returns a numeric vector with the TASW values.

Note

To distribute the computation on more than a cpu

1. install the package 'future'
2. execute in the R session
   - library(future)
   - plan(multisession)

For more options, see the future's documentation

Author(s)

Valentina Masarotto, Guido Masarotto

References

Masarotto, V. & Masarotto, G. (2023) "Covariance-based soft clustering of functional data based on the Wasserstein-Procrustes metric", *Scandinavian Journal of Statistics*, doi:10.1111/sjos.12692.

Examples

```
# Example phoneme.R (simplified) from https://doi.org/10.1111/sjos.12692.
data(phoneme)
# resampling the log-periodograms
# 15 sample covariances for each phoneme
set.seed(12345)
nsubsamples <- 15
n <- 40
gg <- unique(Phoneme)
nphonemes <- length(gg)
N <- n*nsubsamples*nphonemes
M <- NCOL(logPeriodogram)
X <- matrix(NA, N, M)
gr <- integer(N)
r <- 1
first <- 1
last <- n
for (l in gg) {
  for (i in 1:nsubsamples) {
    X[first:last, ] <- logPeriodogram[sample(which(Phoneme==l),n), ]
    gr[first:last] <- r
    r <- r+1
    first <- first+n
    last <- last+n
  }
}
# soft clustering
a <- wassersteinCluster(X, gr)
# how many cluster?
trimmedAverageSilhouette(a)
```

```
# the membership weigths show that the
# algorithm reconstructed the five phoneme
w <- ts(a[[4]]$w)
colnames(w) <- paste("Cluster", 1:5)
plot(w, xlab="Sample covariances", main="")
```

`wassersteinTest` *A permutation or bootstrap test based on optimal transport maps.*

Description

The main function performs a k-sample permutation- or bootstrap-based test to check the equality of covariance operators. More specifically, given a sample of N functional curves belonging to K different populations, each characterized by its own covariance operators, the test aims to check the null hypothesis $\Sigma_1 = \cdots = \Sigma_K$ versus the alternative that at least one operator is different. The test leverages on the equivalence between covariance operators and centered Gaussian processes. In the default version, in order to test the null, the test builds optimal transport maps from the sample to the Wasserstein barycenter of the processes. Successively, it contrasts these maps to the identity operator, as explained in Masarotto, Panaretos & Zemel (2022). However, argument "statistics" allows to base the test directly on the Wasserstein distance between covariance operators, rather than on optimal maps.

Usage

```
wassersteinTest(data, grp, B = 1000,
                statistic = c("transport", "distance"),
                type = c("permutation", "bootstrap"),
                r = c("HS", "trace", "operator"),
                align = TRUE,
                use.future = FALSE,
                iter.bary = 10)
```

Arguments

`data`	A N times M matrix containing the N sample curves; M denotes the number of points of the grid on which the curves are available.
`grp`	Labels that identify which population each curve belongs to.
`B`	Number of permutations or bootstrap replications. If missing, B=1000.
`statistic`	Whether the test is based on the transport maps or directly on the Wasserstein distance. Default is transport.
`type`	Whether the test is permutation or bootstrap based.
`r`	Which norm is used to contrast the test statistics to 0 (used only if statistics="transport"). If r="HS" the Hilbert-Schmidt norm is used, if r="trace" the trace (nuclear) norm is used, if r="operator", the operator norm is used. Default is r="HS".
`align`	If 'align=TRUE', the curves are centered around their mean. Default is TRUE.

`use.future`	Use or not use package 'future' to parallelize the computation? See note.
`iter.bary`	After how many iterations the gradient descent algorithm to compute the barycenter stops.

Value

A list of three returning:

`stat`	Observed value of the test statistics
`p.value`	The p-value indicating the significance level of the test
`trep`	Value of the test statistics for each of the B permutation

Note

To distribute the computation on more than a cpu

1. install the package 'future'
2. execute in the R session
   - library(future)
   - plan(multissession)

For more options, see the future's documentation.

Author(s)

Valentina Masarotto, Guido Masarotto

References

Masarotto, V., Panaretos, V.M. & Zemel, Y. (2022) "Transportation-Based Functional ANOVA and PCA for Covariance Operators", *arXiv*, https://arxiv.org/abs/2212.04797

Examples

```
n = 20
size <- 10
covariances <- rWishart(2,size,diag(size))
A <- covariances[,,1]
B <- covariances[,,2]

# Two groups, each with one covariance. Creates n Gaussian data for each covariance.
# more generally, we could have two groups each with "g_i" covariances in them
g1 <- g2 <- 1
grp <- rep(1:(g1+g2),rep(n,g1+g2))

data <- rbind(matrix(rnorm(n*NCOL(A)),n*g1)%*%A,
             matrix(rnorm(n*NCOL(B)),n*g2)%*%B)
wassersteinTest(data,grp, B=100,r="HS")$p.value

data(phoneme)
wassersteinTest(logPeriodogram, Phoneme, B=100,r="HS")$p.value
```

Index