# Package 'classiMultiFunc'

April 18, 2023

**Title** Classification ensemble models for multivariate functional data

**Version** 0.0.0.9000

**Description** Classification ensemble models for multivariate functional data.

**License** `use_mit_license()`, `use_gpl3_license()` or friends to pick a license

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

## R topics documented:


compmDistMat . . . 2
innerResultsExtract . . . 3
Ker.norm . . . 4
mAitchison . . . 4
mclassiEnsembleLearner . . . 4
mclassiEnsembleTest . . . 5
mclassiEnsembleTrain . . . 5
mclassiGenStack . . . 6
mclassiInested . . . 6
mclassiKernel . . . 7
mclassiKnn . . . 8
mclassiLC . . . 10
mclassiPerf . . . 10
mclassiTies . . . 11
mdcor . . . 11
mdistMeasures . . . 12
mdtw . . . 12
mEuclidean . . . 13
mFrechet . . . 14
mglobMax . . . 15
mglobMaxU . . . 16
mglobMin . . . 17
mglobMinU . . . 18
mHausdorff . . . 19
mkernelChoices . . . 20

mManhattan . . . . . 20
mmean . . . . . 21
mmetricChoices . . . . . 22
mMinkowski . . . . . 22
mnw . . . . . 23
mrange . . . . . 24
mrangeU . . . . . 24
predict.mclassiKernel . . . . . 25
predict.mclassiKnn . . . . . 26
print.mclassiKernel . . . . . 27
print.mclassiKnn . . . . . 27
summary.mclassiKernel . . . . . 27
summary.mclassiKnn . . . . . 27

**Index** **28**

---

`compmDistMat` *compmDistMat*

---

## Description

Distance matrix among $m$-dimensional functions / trajectories

## Usage

```
compmDistMat(
  x,
  y = NULL,
  method = "Euclidean",
  parallel = FALSE,
  cl = NULL,
  diag = TRUE,
  upper = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | `list` of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Values of functions should be of the same time points. |
| `y` | `list` of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($k$) and rows are discrete-time points ($t$). Values of functions should be of the same time points. |
| `method` | `string` to specify the distance measure to compute. Defaults to "Euclidean". The complete list of available options is returned by `mdistMeasures()`. |
| `parallel` | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| `cl` | a cluster object created by parallel. Default is NULL. |

| | |
|---|---|
| diag | is a logical value indicating whether the diagonal of the distance matrix should be printed. Only applicable if distances of one set of functions with itself is computed (isnull(y)=TRUE). If TRUE, diagonal values are NA. |
| upper | is a logical value indicating whether the upper triangle of the matrix should be printed. Only applicable if distances of one set of functions with itself is computed (isnull(y)=TRUE). If TRUE, upper triangle values are NA. |
| ... | other arguments. |

## Details

The present function computes the distance matrix of one set of $n$ different $m$-dimensional functions to another set of $k$ different $m$-dimensional functions. The distances for all combinations of functions in the first set with functions of the second set are returned. Different distance measures are available and can be set by method.

## Value

The present function returns a rectangular matrix of size $k * n$ containing the distances of all $k$ functions in y with all n functions/trajectories in x. If y is left unspecified, the function returns the distances of the functions in x with itself (a symmetric $nxn$ matrix).

## See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel.

---

# innerResultsExtract *innerResultsExtract*

---

## Description

innerResultsExtract

## Usage

```
innerResultsExtract(x, h)
```

## Arguments

| | |
|---|---|
| x | vector |
| h | vector |

`Ker.norm` *Ker.norm computes the values of the normal kernel. Caps values from below at 1e-40, so that no divisions by 0 occur*

## Description

Ker.norm computes the values of the normal kernel. Caps values from below at 1e-40, so that no divisions by 0 occur

## Usage

```
Ker.norm(x)
```

## Arguments

| | |
|---|---|
| `x` | is the entry for Ker.norm function. |

`mAitchison` *Aitchison distance from package robCompositions*

## Description

Aitchison distance from package robCompositions

## Usage

```
mAitchison(data)
```

## Arguments

| | |
|---|---|
| `data` | is the object with the information related to the trajectories, i.e., number of times the mouse was in each of the areas of interest. |

`mclassiEnsembleLearner` *mclassiEnsemble*

## Description

mclassiEnsemble

## Usage

```
mclassiEnsembleLearner(cl, task, id, knn, h = NULL, par.vals, subset)
```

### Arguments

| | |
|---|---|
| cl: | classifier |
| task: | define a task with makeClassifTask |
| id: | name of semi-metrics |
| knn: | value for nearest neighbor |
| h: | Default NULL. If not, bandwidth parameter for the kernel method |
| par.vals: | hyperparameters including metric name, mdist, kernel type, predict.type, set.seed (for ties) |
| subset: | ith inner sample |

---

## mclassiEnsembleTest *mclassiEnsembleTest*

---

### Description

mclassiEnsembleTest

### Usage

```
mclassiEnsembleTest(learner, model, task)
```

### Arguments

| | |
|---|---|
| learner: | learner of class EnsembleLearner |
| model: | model of class EnsembleTrain |
| task: | define a task with makeClassifTask |

---

## mclassiEnsembleTrain *mclassiEnsembleTrain*

---

### Description

mclassiEnsembleTrain

### Usage

```
mclassiEnsembleTrain(learner, task, subset, cv = 5, set.seed = NULL)
```

### Arguments

| | |
|---|---|
| learner: | learner of class EnsembleLearner |
| task: | define a task with makeClassifTask |
| subset: | ith inner sample |
| cv: | splits in the k-fold cross-validation in the inner loop. By default, 5-fold cross-validation is considered |
| set.seed: | seed for the splits in the inner-loop |

## mclassiGenStack *mclassiGenStack*

### Description

mclassiGenStack

### Usage

```
mclassiGenStack(model, prediction, par.vals.ensemble, xreg = NULL)
```

### Arguments

| | |
|---|---|
| model: | model of class mclassiEnsembleTest |
| pred: | predictions of an object of class mclassiEnsembleTest |
| par.vals.ensemble: | list including super.learner (randomForest or boosting), iters for the k-fold cross-validation related to the super.learner, and par.vals.ensemble setting of parameter candidates to be tuned using k-fold cross-validation, and set.seed for the splits of the k-fold cross-valdation to tune the super.learner |
| xreg: | if additional covariates are considered. Default is NULL. |

## mclassiInested *mclassiNestedCV*

### Description

mclassiInested

### Usage

```
mclassiInested(
  cl,
  task,
  id,
  knn,
  h = NULL,
  par.vals,
  par.vals.ensemble = NULL,
  subset,
  cv = 5,
  set.seed = NULL,
  xreg = NULL
)
```

### Arguments

| | |
|---|---|
| cl: | classifier |
| task: | define a task with makeClassifTask |
| id: | name of semi-metrics |
| knn: | value for nearest neighbor |
| h: | value for kernel-based method |
| par.vals: | hyperparameters |
| par.vals.ensemble: | hyperparameters for either random forest or gradient boosting |
| subset: | ith inner sample |
| cv: | splits in the k-fold cross-validation in the inner loop. By default, 5-fold cross-validation is considered |
| set.seed: | seed for the splits in the inner-loop |
| xreg: | if covariates are further considered |

---

`mclassiKernel` *mclassiKernel*

---

### Description

Create mclassiKernel Object

### Usage

```
mclassiKernel(
  classes,
  fdata,
  mdist = NULL,
  metric = "Euclidean",
  kernel = "Ker.norm",
  h = 1,
  nderiv = 0L,
  parallel = FALSE,
  cl = NULL,
  diag = FALSE,
  upper = FALSE,
  ...
)
```

### Arguments

| | |
|---|---|
| classes | `factor` or `numeric`. A vector containing the true classes of the training data. |
| fdata | the training covariates as a `list` of $m$ objects in `matrix` form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Values of functions should be of the same time points. |
| mdist | `matrix`. A customized matrix of distances. Default is NULL. |

| | |
|---|---|
| `metric` | `string` specifying the metric for knn. The complete list of available metrics is returned by `mdistMeasures()`. |
| `kernel` | `string` indicates the type of kernel function to be used. Default is Ker.norm. |
| `h` | `numeric` bandwith parameter. Default is 1. |
| `nderiv` | `integer` number of derivatives of the trajectory. |
| `parallel` | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| `cl` | a cluster object created by parallel. Default is NULL. |
| `diag` | is a logical value indicating whether the diagonal of the distance matrix should be printed. Only applicable if distances of one set of functions with itself is computed (`isnull(y)=TRUE`). If TRUE, diagonal values are NA. |
| `upper` | is a logical value indicating whether the upper triangle of the matrix should be printed. Only applicable if distances of one set of functions with itself is computed (`isnull(y)=TRUE`). If TRUE, upper triangle values are NA. |
| `...` | other arguments. |

## Details

The present function creates an object to specify the kernel-based nonparametric curves discrimination for $n$-different observations of $m$-dimensional functionnal data.

## Value

Returns a list containing the parameters and the training data for a mknn prediction. This list is used for prediction with the function `predict.mclassiKnn`

## See Also

See `makeCluster`, `clusterExport`, `stopCluster`, `parApply` and `parLapply` from parallel

---

`mclassiKnn` *mclassiKnn*

---

## Description

Create mclassiKnn Object

## Usage

```
mclassiKnn(
  classes,
  fdata,
  mdist = NULL,
  metric = "Euclidean",
  knn = 1L,
  nderiv = 0L,
  parallel = FALSE,
  cl = NULL,
  diag = FALSE,
```

```
  upper = FALSE,
  set.seed = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| classes | factor or numeric. A vector containing the true classes of the training data. |
| fdata | the training covariates as a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Values of functions should be of the same time points. |
| mdist | matrix. A customized matrix of distances. Default is NULL. |
| metric | string specifying the metric for knn. The complete list of available metrics is returned by mdistMeasures(). |
| knn | integer specifying the number of nearest neighbours considered in the knn algorithm. |
| nderiv | integer number of derivatives of the trajectory. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |
| diag | is a logical value indicating whether the diagonal of the distance matrix should be printed. Only applicable if distances of one set of functions with itself is computed (isnull(y)=TRUE). If TRUE, diagonal values are NA. |
| upper | is a logical value indicating whether the upper triangle of the matrix should be printed. Only applicable if distances of one set of functions with itself is computed (isnull(y)=TRUE). If TRUE, upper triangle values are NA. |
| set.seed | if a seed wants to be specified. Default is NULL. |
| ... | other arguments. |

## Details

The present function creates an object to specify the k-nearest-neighbor classification for $n$-different observations of $m$-dimensional functionnal data.

## Value

Returns a list containing the parameters and the training data for a mknn prediction. This list is used for prediction with the function predict.mclassiKnn

## See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel

mclassiLC *mclassiCL*

## Description

mclassiCL

## Usage

```
mclassiLC(model, prediction, par.vals.ensemble)
```

## Arguments

| | |
|---|---|
| model: | model of class mclassiEnsembleTest |
| pred: | predictions of an object of class mclassiEnsembleTest |
| par.vals.ensemble: | parameters needed for least squares minimization with function lsei from package limSolve |

mclassiPerf *mclassiPerf*

## Description

mclassiPerf

## Usage

```
mclassiPerf(
  cl,
  task,
  id,
  knn,
  h = NULL,
  par.vals,
  par.vals.ensemble = NULL,
  subset,
  cv = 5,
  set.seed = NULL,
  xreg = NULL
)
```

## Arguments

| | |
|---|---|
| cl: | classifier |
| task: | define a task with makeClassifTask |
| id: | name of semi-metrics |
| knn: | value for nearest neighbor |

| | |
|---|---|
| h: | value for kernel-based method |
| par.vals: | hyperparameters |
| par.vals.ensemble: | hyperparameters for either random forest or gradient boosting |
| subset: | ith inner sample |
| set.seed: | seed for the splits in the inner-loop |
| xreg: | if additional covariates are considered. Default NULL |

---

mclassiTies *mclassiTies*

---

## Description

Provides the positions of the indices of the k nearest trajectories. Selects randomly among ties for the $k$th nearest trajectory.

## Usage

```
mclassiTies(x, knn, set.seed = NULL, ...)
```

## Arguments

| | |
|---|---|
| x | vector of distances of a new trajectory to all training trajectories. |
| knn | number of nearest neighbours. |
| ... | other arguments. |
| if | a seed wants to be specified. |

---

mdcor *mdcor*

---

## Description

mdcor distance between multivariate trajectories

## Usage

```
mdcor(data, parallel = FALSE, cl = NULL, index = 1)
```

## Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Values of functions should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

## Details

Computes the dcor distance for all pairs of $m$-dimensional trajectories.

## mdistMeasures *mdistMeasures*

### Description

Distance between measures of 2-Dimensional trajectories.

### Usage

```
mdistMeasures(data, measure)
```

### Arguments

| | |
|---|---|
| data | list containing one vector (or two if measure="flips") of measures computed by the function mmeasures. |
| measure | string to specify the measure contained in data. The complete list of available measures can be retrieved with the function mmeasuresChoices. For a detailed explanation of the different measures see mousetrap:mt_measures. |

### Details

The mousetrap package enables the calculation of certain measures of mouse trajectories. The present function enables the computation of "distances" between such measures.

## mdtw *mdtw*

### Description

Dynamic Time Warping distance between multivariate functions

### Usage

```
mdtw(data, parallel = FALSE, cl = NULL, ...)
```

### Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Values of functions should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |
| ... | additional arguments that can be passed to dtw::dtw |

## Details

Computes the Dynamic Time Warping distance for all pairs of $m$-dimensional functions.

Applies the `dtw::dtw` function to all pairs of $m$-dimensional functions. By default, the Euclidean Distance between optimally aligned $m$-dimensional functions is computed. Other options and parameters that can be passed to `dtw::dtw` (such as the use of the Manhattan distance) can be defined in the additional parameters ...

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional Dynamic Time Warp distances.

## References


1. Toni Giorgino. *Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package.* Journal of Statistical Software, 31(7), 1-24. doi:10.18637/jss.v031.i07
2. Tormene, P.; Giorgino, T.; Quaglini, S. & Stefanelli, M. *Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation.* Artif Intell Med, 2009, 45, 11-34. doi:10.1016/j.artmed.2008.11.007
3. Sakoe, H.; Chiba, S., *Dynamic programming algorithm optimization for spoken word recognition,* Acoustics, Speech, and Signal Processing, IEEE Transactions on , vol.26, no.1, pp. 43-49, Feb 1978. doi:10.1109/TASSP.1978.1163055
4. Mori, A.; Uchida, S.; Kurazume, R.; Taniguchi, R.; Hasegawa, T. & Sakoe, H. *Early Recognition and Prediction of Gestures* Proc. 18th International Conference on Pattern Recognition ICPR 2006, 2006, 3, 560-563 doi:10.1109/ICPR.2006.467
5. Sakoe, H. *Two-level DP-matching–A dynamic programming-based pattern matching algorithm for connected word recognition* Acoustics, Speech, and Signal Processing, IEEE Transactions on, 1979, 27, 588-595 doi:10.1109/TASSP.1979.1163310
6. Rabiner L, Rosenberg A, Levinson S (1978). *Considerations in dynamic time warping algorithms for discrete word recognition.* IEEE Trans. Acoust., Speech, Signal Process., 26(6), 575-582. doi:10.1109/TASSP.1978.1163164
7. Muller M. *Dynamic Time Warping* in *Information Retrieval for Music and Motion*. Springer Berlin Heidelberg; 2007. p. 69-84. doi:10.1007/9783540740483_4


## See Also

See `makeCluster`, `clusterExport`, `stopCluster`, `parApply` and `parLapply` from parallel, `dist` from proxy

---

`mEuclidean` *mEuclidean*

---

## Description

Euclidean distance between multivariate functions

## Usage

```
mEuclidean(data, parallel = FALSE, cl = NULL)
```

### Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

### Details

Computes the Euclidean distance for all pairs of $m$-dimensional functions. For a single pair of functions, the present R function computes the sum of Euclidean distances between the function values at equal time points.

### Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional Euclidean distances.

### See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel, and dist from proxy

### Examples

```
## 2-dimensional functions

x = replicate(4, rnorm(100, 0, 3))
y = replicate(4, rnorm(100, 3, 1))
data = list(x, y)
meuclidean(data, parallel = FALSE, cl = NULL)

## 3-dimensional functions

z = replicate(4, rpois(100, 2))
data = list(x, y, z)
meuclidean(data, parallel = FALSE, cl = NULL)
```

---

## mFrechet *mFrechet*

---

### Description

mFrechet

### Usage

```
mFrechet(data, parallel = FALSE, cl = NULL, testLeash = -1)
```

mglobMax *mglobMax*

## Description

Global Maximum distance between multivariate functions

## Usage

```
mglobMax(data, parallel = FALSE, cl = NULL)
```

## Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

## Details

Computes the Global Maximum distance for all pairs of $m$-dimensional functions. For a single pair of functions, the present R function returns the maximum euclidean distance between the function values at equal time points.

For each pair of functions f and g, the present R function computes: $maxt[Euclidean_Distance(f(t), g(t))]$

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional global maximum distances.

## See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel, and dist from proxy

## Examples

```
## 2-dimensional functions

x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) )
y = replicate ( 4 , rnorm ( 100 , 3 , 1 ) )
data = list ( x , y )
mglobMax ( data , parallel = FALSE , cl = NULL )

## 3-dimensional functions

z = replicate ( 4 , rpois ( 100 , 2 ) )
data = list ( x , y , z )
mglobMax ( data , parallel = FALSE , cl = NULL )
```

mglobMaxU *mglobMaxU*

## Description

Global Maximum distance between univariate functions

## Usage

```
mglobMaxU(data, parallel = FALSE, cl = NULL)
```

## Arguments

| | |
|---|---|
| data | a matrix that stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

## Details

Computes the Global Maximum distance for all pairs of $m$-dimensional functions. For a single pair of functions, the present R function returns the maximum euclidean distance between the function values at equal time points.

For each pair of functions f and g, the present R function computes: $maxt[Euclidean_Distance(f(t), g(t))]$

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional global maximum distances.

## See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel, and dist from proxy

## Examples

```
## 2-dimensional functions

x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) )
mglobMaxU ( x , parallel = FALSE , cl = NULL )
```

mglobMin *mglobMin*

## Description

Global Minimum distance between multivariate functions

## Usage

```
mglobMin(data, parallel = FALSE, cl = NULL)
```

## Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

## Details

Computes the Global Minimum distance for all pairs of $m$-dimensional functions. For a single pair of functions, the present R function returns the minimum euclidean distance between the function values at equal time points.

For each pair of functions f and g, the present R function computes: $maxt[Euclidean_Distance(f(t), g(t))]$

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional global maximum distances.

## See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel, and dist from proxy

## Examples

```
## 2-dimensional functions

x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) )
y = replicate ( 4 , rnorm ( 100 , 3 , 1 ) )
data = list ( x , y )
mglobMin ( data , parallel = FALSE , cl = NULL )

## 3-dimensional functions

z = replicate ( 4 , rpois ( 100 , 2 ) )
data = list ( x , y , z )
mglobMin ( data , parallel = FALSE , cl = NULL )
```

mglobMinU *mglobMinU*

## Description

Global Minimum distance between univariate functions

## Usage

```
mglobMinU(data, parallel = FALSE, cl = NULL)
```

## Arguments

| | |
|---|---|
| data | a matrix that stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

## Details

Computes the Global Minimum distance for all pairs of $m$-dimensional functions. For a single pair of functions, the present R function returns the minimum euclidean distance between the function values at equal time points.

For each pair of functions f and g, the present R function computes: $maxt[Euclidean_Distance(f(t), g(t))]$

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional global maximum distances.

## See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel, and dist from proxy

## Examples

```
## 2-dimensional functions

x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) )
mglobMinU ( x , parallel = FALSE , cl = NULL )
```

mHausdorff *mHausdorff*

## Description

Hausdorff distance between multivariate functions

## Usage

```
mHausdorff(data, parallel = FALSE, cl = NULL)
```

## Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

## Details

Computes the Hausdorff distance for all pairs of $m$-dimensional functions.

Applies `pracma::hausdorff_dist` to all pairs of $m$-dimensional functions

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional Hausdorff distances.

## See Also

See `makeCluster`, `clusterExport`, `stopCluster`, `parApply` and `parLapply` from parallel, `dist` from proxy

## Examples

```
## 2-dimensional functions

x = replicate(4, rnorm(100, 0, 3))
y = replicate(4, rnorm(100, 3, 1))
data = list(x, y)
mhausdorff(data, parallel = FALSE, cl = NULL)

## 3-dimensional functions

z = replicate(4, rpois(100, 2))
data = list(x, y, z)
mhausdorff(data, parallel = FALSE, cl = NULL)
```

## mkernelChoices *mkernelChoices*

### Description

mkernelChoices

### Usage

```
mkernelChoices()
```

## mManhattan *mManhattan*

### Description

Manhattan distance between multivariate functions

### Usage

```
mManhattan(data, parallel = FALSE, cl = NULL)
```

### Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

### Details

Computes the Manhattan distance between $m$-dimensional functions of the same length. For a single pair of functions, the R function computes the sum of Manhattan distances between the function values at equal time points.

For each pair of functions f and g, the present R function computes: $sumt[Manhattan_distance(f(t), g(t))]$

### Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional Manhattan distances.

### See Also

See makeCluster, clusterExport, stopCluster, parApply and parLapply from parallel, and dist from proxy

## Examples

```
## 2-dimensional functions

x = replicate(4, rnorm(100, 0, 3))
y = replicate(4, rnorm(100, 3, 1))
data = list(x, y)
mmanhattan(data, parallel = FALSE, cl = NULL)

## 3-dimensional functions

z = replicate(4, rpois(100, 2))
data = list(x, y, z)
mmanhattan(data, parallel = FALSE, cl = NULL)
```

---

mmean *mmean*

---

## Description

Mean distance between multivariate functions

## Usage

```
mmean(data, parallel = FALSE, cl = NULL)
```

## Arguments

| | |
|---|---|
| data | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| parallel | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| cl | a cluster object created by parallel. Default is NULL. |

## Details

Computes the Mean distance for all pairs of $m$-dimensional functions.

For each pair of functions f and g, the present R function computes: $Euclidian_distance(mean(f(t)), mean(g(t))]$

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional Mean distances.

## See Also

See `makeCluster`, `clusterExport`, `stopCluster`, `parApply` and `parLapply` from parallel, and `dist` from proxy

## Examples

```
## 2-dimensional functions

x = replicate(4, rnorm(100, 0, 3))
y = replicate(4, rnorm(100, 3, 1))
data = list(x, y)
mmean(data, parallel = FALSE, cl = NULL)

## 3-dimensional functions

z = replicate(4, rpois(100, 2))
data = list(x, y, z)
mmean(data, parallel = FALSE, cl = NULL)
```

---

`mmetricChoices` *mmetricChoices*

---

## Description

mmetricChoices

## Usage

```
mmetricChoices(proxy.only = FALSE)
```

---

`mMinkowski` *mMinkowski*

---

## Description

Minkowski distance between multivariate functions

## Usage

```
mMinkowski(data, p = 2, parallel = FALSE, cl = NULL)
```

## Arguments

| | |
|---|---|
| `data` | a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points. |
| `p` | integer to specify which p-norm to compute |
| `parallel` | logical value indicating whether computations should be parallelized. Default is FALSE. If TRUE, parallelization is conducted with parallel package. |
| `cl` | a cluster object created by parallel. Default is NULL. |

## Details

Computes the Minkowski distance for all pairs of $m$-dimensional functions. For a single pair of functions, the present R function computes the sum of Minkowski distances between the function values at equal time points.

For each pair of functions f and g, the present R function computes: $sumt[Minkowski_distance(f(t), g(t))]$

## Value

Returns a square and symmetric $nxn$ matrix of $m$-dimensional Minkowski distances.

## See Also

See `makeCluster`, `clusterExport`, `stopCluster`, `parApply` and `parLapply` from parallel, and `dist` from proxy

## Examples

```
## 2-dimensional functions

x = replicate(4, rnorm(100, 0, 3))
y = replicate(4, rnorm(100, 3, 1))
data = list(x, y)
mminkowski(data, parallel = FALSE, cl = NULL)

## 3-dimensional functions

z = replicate(4, rpois(100, 2))
data = list(x, y, z)
mminkowski(data, parallel = FALSE, cl = NULL)
```

---

`mnw` *mnw*

---

## Description

Needleman Wunsch

## Usage

```
mnw(data, parallel = FALSE, cl = NULL, method = "levenshtein")
```

mrange *mrange*

## Description

Range distance between multivariate trajectories

## Usage

```
mrange(data, parallel = FALSE, cl = NULL)
```

## Details

Computes the Range distance for all pairs of $m$-dimensional trajectories For a single pair of functions, the current R function returns the absolute difference between the ranges of $m$-dimensional trajectories.

x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) ) y = replicate ( 4 , rnorm ( 100 , 5 , 3 ) ) data = list ( x = x, y = y ) mrange ( data , parallel = FALSE , cl = NULL )

mrangeU *mrangeU*

## Description

Range distance between multivariate trajectories

## Usage

```
mrangeU(data, parallel = FALSE, cl = NULL)
```

## Details

Computes the Range distance for all pairs of $m$-dimensional trajectories For a single pair of functions, the current R function returns the absolute difference between the ranges of $m$-dimensional trajectories.

x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) ) mrangeU ( x , parallel = FALSE , cl = NULL )

predict.mclassiKernel *predict.mclassiKernel*

## Description

Predictions with an mclassiKernel object

## Usage

```
## S3 method for class 'mclassiKernel'
predict(object, newdata = NULL, Mdist = NULL, predict.type = "response")
```

## Arguments

| | |
|---|---|
| object | mclassiKernel object defined with mclassiKernel |
| newdata | the functional covariates for prediction as a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points as the training data. |
| predict.type | string to specify type of prediction. Either "response" or "prob" |

## Details

The present function yields predictions for new data with an mclassiKernel object .

## Examples

```
## 3-dimensional functions
#classes
classes = as.factor ( c ( "Cat" , "Dog" , "Dog" , "Cat" ) )

#training_data
x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) )
y = replicate ( 4 , rnorm ( 100 , 3 , 1 ) )
z = replicate ( 4 , rpois ( 100 , 2 ) )
training_data = data.frame ( cbind ( id = 1:4 , dim = 3 , t ( x ), t ( y ) , t ( z ) ) )

#Creating object for mclassiKernel prediction
object = mclassiKernel ( classes = classes , fdata = training_data , h = 30 , kernel = "Ker.norm" , nderiv = 0 , cl

#test_data
x = replicate (2, rnorm ( 100 , 0 , 2.9 ) )
y = replicate (2, rnorm ( 100 , 3 , 1.5 ) )
z = replicate (2, rpois ( 100 , 3 ) )
test_data = data.frame ( cbind ( id = 5:6 , dim = 3 , t ( x ), t ( y ) , t ( z ) ) )

#Predict

predict.mclassiKernel ( object = object, newdata= test_data, predict.type = "response" )
```

predict.mclassiKnn *predict.mclassiKnn*

## Description

Predictions with an mclassiKnn object

## Usage

```
## S3 method for class 'mclassiKnn'
predict(object, newdata = NULL, Mdist = NULL, predict.type = "response")
```

## Arguments

| | |
|---|---|
| object | mknn object defined with mclassiKnn |
| newdata | the functional covariates for prediction as a list of $m$ objects in matrix form. Each matrix stores a dimension of the set of functions, such that columns are individuals ($n$) and rows are discrete-time points ($t$). Functions' values should be of the same time points as the training data. |
| predict.type | string to specify type of prediction. Either "response" or "prob" |

## Details

The present function yields predictions for new data with an mclassiKnn object .

## Examples

```
## 3-dimensional functions
#classes
classes = as.factor ( c ( "Cat" , "Dog" , "Dog" , "Cat" ) )

#fdata
x = replicate ( 4 , rnorm ( 100 , 0 , 3 ) )
y = replicate ( 4 , rnorm ( 100 , 3 , 1 ) )
z = replicate ( 4 , rpois ( 100 , 2 ) )
training_data = data.frame ( cbind ( id = 1:4 , dim = 3 , t ( x ), t ( y ) , t ( z ) ) )

#Creating object for mclassiKnn prediction
object = mclassiKnn ( classes = classes , fdata = training_data , knn=3 , nderiv = 0 , cl = NULL )

#test_data
x = replicate (2, rnorm ( 100 , 0 , 2.9 ) )
y = replicate (2, rnorm ( 100 , 3 , 1.5 ) )
z = replicate (2, rpois ( 100 , 3 ) )
test_data = data.frame ( cbind ( id = 5:6 , dim = 3 , t ( x ), t ( y ) , t ( z ) ) )

#Predict

predict.mclassiKnn ( object = object , newdata= test_data , predict.type = "response" )
```

## print.mclassiKernel *print.mclassiKernel*

### Description

print.mclassiKernel

### Usage

```
## S3 method for class 'mclassiKernel'
print(x, ...)
```

## print.mclassiKnn *print.mclassiKnn*

### Description

print.mclassiKnn

### Usage

```
## S3 method for class 'mclassiKnn'
print(x, ...)
```

## summary.mclassiKernel *summary.mclassiKernel*

### Description

summary.mclassiKernel

### Usage

```
## S3 method for class 'mclassiKernel'
summary(object, ...)
```

## summary.mclassiKnn *summary.mclassiKnn*

### Description

summary.mclassiKnn

### Usage

```
## S3 method for class 'mclassiKnn'
summary(object, ...)
```

# Index


clusterExport, *3*, *8, 9*, *13–21*, *23*
compmDistMat, 2

dist, *13–21*, *23*
dtw::dtw, *12, 13*

innerResultsExtract, 3

Ker.norm, 4

mAitchison, 4
makeCluster, *3*, *8, 9*, *13–21*, *23*
mclassiEnsembleLearner, 4
mclassiEnsembleTest, 5
mclassiEnsembleTrain, 5
mclassiGenStack, 6
mclassiInested, 6
mclassiKernel, 7, *25*
mclassiKnn, 8, *26*
mclassiLC, 10
mclassiPerf, 10
mclassiTies, 11
mdcor, 11
mdistMeasures, 12
mdtw, 12
mEuclidean, 13
mFrechet, 14
mglobMax, 15
mglobMaxU, 16
mglobMin, 17
mglobMinU, 18
mHausdorff, 19
mkernelChoices, 20
mManhattan, 20
mmean, 21
mmeasures, *12*
mmeasuresChoices, *12*
mmetricChoices, 22
mMinkowski, 22
mnw, 23
mousetrap:mt_measures, *12*
mrange, 24
mrangeU, 24

parApply, *3*, *8, 9*, *13–21*, *23*
parLapply, *3*, *8, 9*, *13–21*, *23*
pracma::hausdorff_dist, *19*
predict.mclassiKernel, 25
predict.mclassiKnn, *8, 9*, 26
print.mclassiKernel, 27
print.mclassiKnn, 27

stopCluster, *3*, *8, 9*, *13–21*, *23*
summary.mclassiKernel, 27
summary.mclassiKnn, 27