# Package ‘ORION’

May 15, 2020

**Type** Package

**Title** Ordinal Relations

**Version** 1.0.1

**Date** 2020-05-12

**Author** L Lausser, LM Schaefer, HA Kestler

**Maintainer** HA Kestler <hans.kestler@uni-ulm.de>

**Description** Functions to handle ordinal relations reflected within the feature space. Those function allow to search for ordinal relations in multi-class datasets. One can check whether proposed relations are reflected in a specific feature representation. Furthermore, it provides functions to filter, organize and further analyze those ordinal relations.

**License** GPL-2

**LazyLoad** yes

**Imports** e1071,TunePareto, knitr, rmarkdown, doParallel, igraph, foreach, randomForest

**NeedsCompilation** yes

**VignetteBuilder** knitr

**RoxygenNote** 7.1.0

**Repository** CRAN

**Date/Publication** 2020-05-15 15:10:06 UTC

## R topics documented:

## `as.edgedataframe` *Coerce to an Edge List*

### Description

Converts from a Subcascades object to a weighted edge list.

### Usage

```
as.edgedataframe(subcascades)
```

### Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |

### Details

Converts a Subcascades object to a data frame that can be used to generate a graph. The first and second column correspond to all pairwise relations of the cascades within the Subcascades object. The 'CASC_ID' column contains the same ID for all edges belonging to the same cascade. The 'SIZE' column gives the size of the cascade to which the repective column bleongs.

## Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

## See Also

as.groupwise, as.subcascades

## Examples

```
library(TunePareto)
library(igraph)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')

# generate a dataframe
subcascades = subcascades(genMap,thresh=0.65,size=4)
edges = as.edgedataframe(subcascades)
g = graph_from_data_frame(edges[,c(1,2)], directed = TRUE)
E(g)$weight = edges[,3]
plot(g,edge.color=edges[,3],edge.arrow.size=0.5,
edge.curved =seq(-0.5, 1, length = ecount(g)))
```

---

# as.groupwise *Coerce to a Groupwise object*

---

## Description

Converts from a Subcascades object to a Groupwise object.

## Usage

```
as.groupwise(subcascades = NULL, maxCl = 50)
```

## Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| `maxCl` | An integer defining the lower bound for the maximal number of classes. Has only to be set if the analyzed dataset has more than 50 classes. |

## Details

This function re-sorts the Subcascades object in a way that the cascades made up of the same classes are grouped.

## Value

A Groupwise object, which comprises a two-leveled list. The first level collects cascades of the same size. The second level contains a list of unique class combinations, labelled as a character string with '-' separating the different classes. For each unique set of class combinations the corresponding orders and their performance are given as a matrix, where each row contains a cascade, given as a character string of type '1>2>3', and the columns the sensitivity for the class at the corresponding position. Each matrix is sorted row-wise according to the achieved minimal classwise sensitivites of the cascades (decreasing).

## See Also

`as.subcascades`, `summaryGroupwise`, `as.edgedataframe`

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.7)
```

## as.subcascades — *Coerce to a Subcascades Object*

### Description

Converts from a Groupwise object to a Subcascades object.

### Usage

```
as.subcascades(groupwise = NULL)
```

### Arguments

| | |
|---|---|
| `groupwise` | A Groupwise object, which comprises a two-leveled list. The first level collects cascades of the same size. The second level contains a list of unique class combinations, labelled as a character string with '-' separating the different classes. For each unique set of class combinations the corresponding orders and their performance are given as a matrix, where each row contains a cascade, given as a character string of type '1>2>3', and the columns the sensitivity for the class at the corresponding position. Each matrix is sorted row-wise according to the achieved minimal classwise sensitivites of the cascades (decreasing). |

### Details

Converts a Groupwise object to a Subcascades object.

### Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

### See Also

`as.groupwise`, `as.edgedataframe`

### Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')

# generate a Groupwise object
subcascades = subcascades(genMap,thresh=0.7)
groupwise = as.groupwise(subcascades)

#convert it so a Subcascades object
as.subcascades(groupwise)
```

## confusion.table *Calculation of a Confusion Table*

### Description

Confusion table and class assignment of one cascade.

### Usage

```
confusion.table(
  predictionMap = NULL,
  cascade = NULL,
  other.classes = NULL,
  sort = TRUE
)
```

### Arguments

| | |
|---|---|
| predictionMap | A PredictionMap object as it is returned by gen.predictionMap-function. It is made up of two elements(pred and meta). The rownames of the pred-matrix (e.g. [0vs1]) show the classes of the binary base classifier. The elements are the prediction result of a specific training. The rows that correspond to base classifiers that would separate the same class consists of -1. Those rows are not used within the analysis. The meta information connects the values in the pred-matrix to a specific fold, run and contains the original label. |
| cascade | A numeric vector of classes or a character string of type '1>2>3' of at least two class labels reflected in 'predictionMap'. |
| other.classes | This parameter can be either NULL, 'all' or a numeric vector of classes that are not part of the cascade parameter. If other.classes is: • NULL, only the cascade classes are evaluated. • 'all', all remaining classes are evaluated. • a vector of classes, those classes are evaluated. |
| sort | If TRUE (default) the classes that are not part of cascade are sorted based on their confusion. |

## Value

A confusion matrix of sensitivities, with the label of the predicted classes in the rows and the labels of the original class in the columns.

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')

# Calculation of the confusion matrix for '0>2>3>4'.
confusion.table(genMap, cascade = '0>2>3>4')
# Calculation of the confusion matrix for '0>2>3>4'
# and the assignment of all samples of the other classes.
confusion.table(genMap, cascade = '0>2>3>4',
                other.classes='all', sort = TRUE)
```

---

`dropSets` *Filter out Subcascades*

---

## Description

`dropSets` filters out specific cascade sets.

## Usage

```
dropSets(
  subcascades = NULL,
  sets = NULL,
  direction = "sub",
  ordered = FALSE,
  neighborhood = "direct",
  type = "any"
)
```

## Arguments

`subcascades` A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to

| | |
|---|---|
| | the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| `sets` | Contains the set used for filtering. It is either a list of numberic vectors, a numeric vector, or a vector of characters representing a cascade of the following format '1>2>4'. |
| `direction` | Either 'sub','super' or 'exact', indicating whether the subset, the superset or the exact set is filtered. The 'exact' case filters exactly the cascades defined in `sets`. Please refer to the details section for a detailed description. |
| `ordered` | Either TRUE or FALSE indicating whether the order of the classes in sets is considered or not. |
| `neighborhood` | Either 'direct' or 'indirect' defines whether the given classes have to be direct neighbors ('direct') or there are other classes allowed inbetween ('indirect'). |
| `type` | If more than one cascade is defined in sets, this parameter defines whether the filtered cascade has to fit to at least one of the sets ('any') or to all of the given sets ('all'). |

## Details

This function allows for filtering the Subcascades object.

If `direction` is set to 'exact' the parameter `neighborhood` is ignored and the parameter `type` is used as its default and cannot be changed. There has to be an exact match between the classes of the sets parameter cascades and the cascade of the Subcascades object, so not more but also not less classes are allowed. If the `ordered` parameter is set to TRUE also the order of the classes within the `sets` parameter and within the Subcascades object has be be exactly the same.

If `direction` is set to 'sub' the Subcascades object is filtered for subsets of the given cascades. If the `type` parameter is set to 'any' each cascade is considered individually otherwise all cascades are used as reference for filtering. This means that either for each cascade of the `sets` parameter individually or for taking all together the Subcascades object is filtered for cascades that are made up of the same classes, a subset of classes (`ordered = FALSE`) or a cascade, part of a cascade (`ordered = TRUE`), resulting in a set of cascades that might contain less classes. Depending on the parameter `neighborhood` each class has to have the same direct neighbors as in the `sets` parameter defined ('direct'). If `neighborhood` is set to 'indirect' the filtering is less strict and the direct neighborhood defined in `sets` is not considered.

If `direction` is set to 'super' the Subcascades object is filtered for a superset of the given cascades. If the `type` parameter is set to 'any' each cascade is considered individually otherwise all cascades are used as reference for filtering. A superset are cascades that are made up of (parts) of the given cascades (`ordered = TRUE`) or classes (`ordered=FALSE`) but can contain also more classes. Depending on the parameter `neighborhood` each class has to have the same direct neighbors as in the `sets` parameter defined ('direct'). If `neighborhood` is set to 'indirect' the filtering is less strict and the direct neighborhood defined in `sets` is not considered.

## Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

## See Also

`dropSize`, `keepSize`, `keepSets`, `dropThreshold`, `keepThreshold`

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.7)

#define sets
set1 = list(c(1,2,3),c(2,3,4))
set2 = c('1>2>3','2>3>4')

# filter for the subset cascades that contain either the classes
# {1,2,3} or {2,3,4} independent of the order, but neighbored
dropSets(subcascades, sets = set1, direction = 'sub',
         ordered = FALSE, neighborhood = 'direct')
dropSets(subcascades, sets = set2, direction = 'sub',
         ordered = FALSE, neighborhood = 'direct')

# filter for the superset cascades that contain either the classes
# {1,2,3} or {2,3,4} independent of the order, but neighbored
dropSets(subcascades, sets = set1, direction = 'super',
         ordered = FALSE, neighborhood = 'direct')
dropSets(subcascades, sets = set2, direction = 'super',
         ordered = FALSE, neighborhood = 'direct')

# filter for the superset cascades that contain both the classes
# {1,2,3} and {2,3,4} in exactly the given order, but allowing
# for other classes inbetween
dropSets(subcascades, sets = set1, direction = 'super',
        ordered = TRUE, neighborhood = 'indirect', type = 'all')
dropSets(subcascades, sets = set2, direction = 'super',
        ordered = TRUE, neighborhood = 'indirect', type = 'all')

# filter for the exact cascades
# sets can be a numeric list
```

```
result <- dropSets(subcascades, list(c(0,1,2),c(2,3,4,1)),
                   direction = 'exact', ordered=TRUE)
unlist(t(lapply(result,rownames)))
# or sets can be a character vector
result <- dropSets(subcascades, c('0>1>2','2>3>4>1'),
                   direction = 'exact',ordered=TRUE)
unlist(t(lapply(result,rownames)))
```

## dropSize *Filters for size*

### Description

Filters out the Subcascades object for the given sizes.

### Usage

```
dropSize(subcascades = NULL, size = NA)
```

### Arguments

| | |
|---|---|
| subcascades | A Subcascades object as it is returned by subcascades-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| size | A numeric value that defines the size of the cascades that should be returned. The smallest size is 2 and the largest the maximal number of classes of the current dataset. |

### Value

A Subcascades object as in subcascades, that does not include cascades of the specific lengths that hve been filtered.

### See Also

keepSize, dropSets, keepSets, dropThreshold, keepThreshold

### Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.7)

# filters out cascades that have a length of 3
dropSize(subcascades,size=3)
# filters out cascades that have a length of 3 or 4
dropSize(subcascades, size=c(3,4))
```

## `dropThreshold` *Exclude Cascades Based on Threshold*

### Description

Filters out all cascades that match the comparison with a minimal classwise sensitivity threshold.

### Usage

```
dropThreshold(subcascades = NULL, comparison = ">=", thresh = 0)
```

### Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| `comparison` | Defines the comparison type (<,>,<=,>=) for the threshold. |
| `thresh` | A numeric value between 0 and 1. The minimal sensitivity threshold used to filter the returned cascades. Only cascades that pass this threshold are returned. If 0 is used the returned cascades are filtered for >0 and otherwise >= thresh. For low thresholds the calculation lasts longer. |

### Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

### See Also

`dropSize`, `keepSize`, `dropSets`, `keepSets`, `keepThreshold`

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.5)

# filters for cascades that
# 1. have a minimal classwise sensitivity >= 0.6
dropThreshold(subcascades,thresh=0.6)
# 2. have a minimal classwise sensitivity <= 0.6
dropThreshold(subcascades, comparison = '<=', thresh=0.6)
```

---

`esl` *ESL Dataset*

---

## Description

The original ESL data set contains 488 profiles of applicants for certain industrial jobs. Expert psychologists of a recruiting company, based upon psychometric test results and interviews with the candidates, determined the values of the input attributes. The output is the an overall score corresponding to the degree of fitness of the candidate to this type of job. Here classes 1-3 are merged to one group and all classes >6 are merged to one group. Furtheremore, the data is relabeled to start at 0 and only 20 samples per class are included.

## Usage

```
data(esl)
```

## Format

An object of class `list` of length 2.

## Source

Weka dataset (datasets-arie_ben_david.tar.gz, 11,348 Bytes)

## References

ordinal, real-world datasets donated by Dr. Arie Ben David (Holon Inst. of Technology/Israel) Donor: Arie Ben David MIS, Dept. of Technology Management Holon Academic Inst. of Technology 52 Golomb St. Holon 58102 Israel abendav@hait.ac.il Owner: Yoav Ganzah Business Administration School Tel Aviv Univerity Ramat Aviv 69978 Israel

## Examples

```
data(esl)
```

---

`esl_org` *ESL Dataset*

---

## Description

The ESL data set contains 488 profiles of applicants for certain industrial jobs. Expert psychologists of a recruiting company, based upon psychometric test results and interviews with the candidates, determined the values of the input attributes. The output is the an overall score corresponding to the degree of fitness of the candidate to this type of job.

## Usage

```
data(esl_org)
```

## Format

An object of class `data.frame` with 488 rows and 5 columns.

## Source

Weka dataset (datasets-arie_ben_david.tar.gz, 11,348 Bytes)

## References

ordinal, real-world datasets donated by Dr. Arie Ben David (Holon Inst. of Technology/Israel) Donor: Arie Ben David MIS, Dept. of Technology Management Holon Academic Inst. of Technology 52 Golomb St. Holon 58102 Israel abendav@hait.ac.il Owner: Yoav Ganzah Business Administration School Tel Aviv Univerity Ramat Aviv 69978 Israel

## Examples

```
data(esl_org)
```

## gen.conf *Calculation of Binary Classifier Sensitivities*

### Description

Sensitivities of all pairwise binary classifiers for all classes.

### Usage

```
gen.conf(predictionMap = NULL)
```

### Arguments

| | |
|---|---|
| `predictionMap` | A PredictionMap object as it is returned by `gen.predictionMap`-function. It is made up of two elements(pred and meta). The rownames of the pred-matrix (e.g. [0vs1]) show the classes of the binary base classifier. The elements are the prediction result of a specific training. The rows that correspond to base classifiers that would separate the same class consists of -1. Those rows are not used within the analysis. The meta information connects the values in the pred-matrix to a specific fold, run and contains the original label. |

### Details

The Conf object contains all class sensitivities for each binary trained classifiers. The $fC-part is a column-vector and contains the sensitivities for the first class of each pairwise classifier. The rows stand for the pairwise classifiers, whereby 0vs1 means that this classifier was trained for class 0 against class 1, with class 0 being the first class. The number '-1' is used as placeholder. The $sC-part is a matrix and contains the preformance measures for all second classes, which are meant here as all classes except the first class. The rows correspond to the binary classifiers and the columns to the classes. Furthermore the foldlist ($foldList) used and the dataset name ($dataName) are saved as variable of the object.

### Value

Object of class Conf. For a detailed description refer to details sections.

### Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')

gen.conf(genMap)
```

## gen.predictionMap *Generation of a Prediction Map Object*

### Description

Makes a PredictionMap object for the given data.

### Usage

```
gen.predictionMap(
  data = NULL,
  labels = NULL,
  foldList = NULL,
  parallel = FALSE,
  classifier = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| `data` | The data either as matrix or dataframe with features in rows and samples in columns. |
| `labels` | A vector of labels of data, consecutively labelled starting with 0. |
| `foldList` | A set of partitions for a cross-validation experiment. This list comprises as many elements as cross-validation runs. Each run is a list of as many vectors as folds. The entries are the indices of the samples that are left out in the folds. This fold list can be generated by using `generateCVRuns`. |
| `parallel` | Either TRUE or FALSE (default). If TRUE the pairwise training is performed parallelized. |
| `classifier` | A TunePareto classifier object.<br>For detailed information refer to `TunePareto::tuneParetoClassifier`. |
| `...` | Further parameters of the classifier object. |

### Details

Using a cross-validation set up this function performs a pairwise training for all class combinations and evaluates all samples using the trained classifier. This means that for each run, fold and binary classifier the predicted class for each sample is calculated.

## Value

A PredictionMap object. It is made up of two elements. The meta information which connects the values in the pred-matrix to a specific fold, run and contains the original label. The rownames of the pred-matrix show the classes of the binary base classifier. The elements are the prediction result of a specific training. The rows that correspond to base classifiers that would separate the same class consists of -1. Those rows are not used within the analysis.

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)

# svm with linear kernel
genMap = gen.predictionMap(data, labels, foldList = foldList,
                           classifier = tunePareto.svm(), kernel='linear')

# knn with k = 3
genMap = gen.predictionMap(data, labels, foldList = foldList,
                           classifier = tunePareto.knn(), k = 3)
# randomForest
genMap = gen.predictionMap(data, labels, foldList = foldList,
                           classifier = tunePareto.randomForest())
```

---

`keepSets` *Filter Subcascades*

---

## Description

`keepSets` filters specific cascade sets.

## Usage

```
keepSets(
  subcascades = NULL,
  sets = NULL,
  direction = "sub",
  ordered = FALSE,
  neighborhood = "direct",
  type = "any"
)
```

## Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| `sets` | Contains the set used for filtering. It is either a list of numberic vectors, a numeric vector, or a vector of characters representing a cascade of the following format '1>2>4'. |
| `direction` | Either 'sub','super' or 'exact', indicating whether the subset, the superset or the exact set is filtered. The 'exact' case filters exactly the cascades defined in `sets`. Please refer to the details section for a detailed description. |
| `ordered` | Either TRUE or FALSE indicating whether the order of the classes in sets is considered or not. |
| `neighborhood` | Either 'direct' or 'indirect' defines whether the given classes have to be direct neighbors ('direct') or there are other classes allowed inbetween ('indirect'). |
| `type` | If more than one cascade is defined in sets, this parameter defines whether the filtered cascade has to fit to at least one of the sets ('any') or to all of the given sets ('all'). |

## Details

This function allows for filtering the Subcascades object.

If `direction` is set to 'exact' the parameter `neighborhood` is ignored and the parameter `type` is used as its default and cannot be changed. There has to be an exact match between the classes of the sets parameter cascades and the cascade of the Subcascades object, so not more but also not less classes are allowed. If the `ordered` parameter is set to TRUE also the order of the classes within the `sets` parameter and within the Subcascades object has be be exactly the same.

If `direction` is set to 'sub' the Subcascades object is filtered for subsets of the given cascades. If the `type` parameter is set to 'any' each cascade is considered individually otherwise all cascades are used as reference for filtering. This means that either for each cascade of the `sets` parameter individually or for taking all together the Subcascades object is filtered for cascades that are made up of the same classes, a subset of classes (`ordered = FALSE`) or a cascade, part of a cascade (`ordered = TRUE`), resulting in a set of cascades that might contain less classes. Depending on the parameter `neighborhood` each class has to have the same direct neighbors as in the `sets` parameter defined ('direct'). If `neighborhood` is set to 'indirect' the filtering is less strict and the direct neighborhood defined in `sets` is not considered.

If `direction` is set to 'super' the Subcascades object is filtered for a superset of the given cascades. If the `type` parameter is set to 'any' each cascade is considered individually otherwise all cascades are used as reference for filtering. A superset are cascades that are made up of (parts) of the given cascades (`ordered = TRUE`) or classes (`ordered=FALSE`) but can contain also more classes. Depending on the parameter `neighborhood` each class has to have the same direct neighbors as in the `sets` parameter defined ('direct'). If `neighborhood` is set to 'indirect' the filtering is less strict and the direct neighborhood defined in `sets` is not considered.

## Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

## See Also

dropSize, keepSize, dropSets, dropThreshold, keepThreshold

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.7)

#define sets
set1 = list(c(1,2,3),c(2,3,4))
set2 = c('1>2>3','2>3>4')

# filter for the subset cascades that contain either the classes
# {1,2,3} or {2,3,4} independent of the order, but neighbored
keepSets(subcascades, sets = set1, direction = 'sub',
         ordered = FALSE, neighborhood = 'direct')
keepSets(subcascades, sets = set2, direction = 'sub',
         ordered = FALSE, neighborhood = 'direct')

# filter for the superset cascades that contain either the classes
# {1,2,3} or {2,3,4} independent of the order, but neighbored
keepSets(subcascades, sets = set1, direction = 'super',
         ordered = FALSE, neighborhood = 'direct')
keepSets(subcascades, sets = set2, direction = 'super',
         ordered = FALSE, neighborhood = 'direct')

# filter for the superset cascades that contain both the classes
# {1,2,3} and {2,3,4} in exactly the given order, but allowing
# for other classes inbetween
keepSets(subcascades, sets = set1, direction = 'super',
        ordered = TRUE, neighborhood = 'indirect', type = 'all')
keepSets(subcascades, sets = set2, direction = 'super',
        ordered = TRUE, neighborhood = 'indirect', type = 'all')
```

```
# filter for the exact cascades
# sets can be a numeric list
result <- keepSets(subcascades, list(c(0,1,2),c(2,3,4,1)),
                   direction = 'exact', ordered=TRUE)
unlist(t(lapply(result,rownames)))
# or sets can be a character vector
result <- keepSets(subcascades, c('0>1>2','2>3>4>1'),
                   direction = 'exact',ordered=TRUE)
unlist(t(lapply(result,rownames)))
```

## `keepSize` *Filter for Size*

### Description

Filters the Subcascades object for cascades of the given sizes.

### Usage

```
keepSize(subcascades = NULL, size = NA)
```

### Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| `size` | A numeric value that defines the size of the cascades that should be returned. The smallest size is 2 and the largest the maximal number of classes of the current dataset. |

### Value

A Subcascades object as in `subcascades`, that only includes cascades of specific lengths.

### See Also

`dropSize`, `dropSets`, `keepSets`, `dropThreshold`, `keepThreshold`

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.7)

# filters for cascades that have a length of 3
keepSize(subcascades,size=3)
# filters for cascades that have a length of 3 or 4
keepSize(subcascades, size=c(3,4))
```

---

`keepThreshold` *Filters for Threshold*

---

## Description

Filters for all cascades that match the comparison with a minimal classwise sensitivity threshold.

## Usage

```
keepThreshold(subcascades = NULL, thresh = 0, comparison = ">=")
```

## Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| `thresh` | A numeric value between 0 and 1. The minimal sensitivity threshold used to filter the returned cascades. Only cascades that pass this threshold are returned. If 0 is used the returned cascades are filtered for >0 and otherwise >= thresh. For low thresholds the calculation lasts longer. |
| `comparison` | Defines the comparison type (<,>,<=,>=) for the threshold. |

## Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

## See Also

`dropSize`, `keepSize`, `dropSets`, `keepSets`, `dropThreshold`

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.5)

# filters for cascades that
# 1. have a minimal classwise sensitivity >= 0.6
keepThreshold(subcascades,thresh=0.6)
# 2. have a minimal classwise sensitivity <= 0.6
keepThreshold(subcascades, comparison = '<=', thresh=0.6)
```

---

`mergeSubcascades` *Merge Subcascades*

---

## Description

`mergeSubcascades` adds the cascades from subcascades2 to the subcascades1, that have not been part of subcascades1.

## Usage

```
mergeSubcascades(subcascades1 = NULL, subcascades2 = NULL)
```

### Arguments

| | |
|---|---|
| `subcascades1` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length. The rownames show the class order and the entries the sensitivity for each position of the cascade. |
| `subcascades2` | A Subcascades object like subcascades1 |

### Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

### Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')

# make two Subcascades objects
subcascades1 = subcascades(genMap, size = c(3,4), thresh = 0.6)
subcascades2 = subcascades(genMap, size = c(4), thresh = 0.5)
# add the cascades of subcascades2 to subcascades1
mergeSubcascades(subcascades1, subcascades2)
```

---

## `plot.Conf` *Base Classifier Performance Heatmap*

---

### Description

Plots a heatmap that shows base classifier performance.

### Usage

```
## S3 method for class 'Conf'
plot(
  x = NULL,
  classNames = NULL,
  symmetric = FALSE,
  main = "summary of base classifiers",
  xlab = "second class",
  ylab = "first class",
  digits = 3,
  ignore = 0,
  first.colors = c("#f5f5f5", "#8c510a"),
  second.colors = c("#f5f5f5", "#01665e"),
  na.color = c("yellow"),
  las = 1,
  color.key = TRUE,
  cex = 1,
  cex.lab = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A Conf object as it is returned by `gen.conf`-function. |
| `classNames` | Vector of the original class names. If not given the class number is used instead. |
| `symmetric` | Logical indicating whether a symmetric base classifier (TRUE) is used. |
| `main` | See `plot`. |
| `xlab` | A title for the x axis (see `plot`). |
| `ylab` | A title for the y axis (see `plot`). |
| `digits` | Integer indicating the number of decimal places to be used (see `round`). |
| `ignore` | A numeric value between 0 and 1. All confusion and purity values below this number are not written as string into the corresponding element. |
| `first.colors` | A 2-element vector of the color for the minimal and maximal class-wise sensitivity of the first class. The color palette is calcuated by an interpolation between the 2 given colors. |
| `second.colors` | A 2-element vector of the color for the minimal and maximal class-wise sensitivity of the second class. The color palette is calcuated by an interpolation between the 2 given colors. |
| `na.color` | Color, which is used for indicating the empty elements (if the given class is not part of the cascade). |
| `las` | See `par`. |
| `color.key` | Specifies whether a color key is drawn (TRUE) or not (FALSE). |
| `cex` | See `par`. |
| `cex.lab` | See `par`. |
| `...` | Further arguments passed from other methods. |

## Details

This function plots a heatmap of the base classifier performance.

## Value

No return value, called to generate the heatmap plot.

## See Also

gen.conf, plot.Subcascades, plot.ConfusionTable

## Examples

```
library(TunePareto)
data(esl)
data <- esl$data
labels <- esl$labels
foldList <- generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap <- gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
conf <- gen.conf(genMap)

plot(conf,symmetric=TRUE)
```

---

## plot.ConfusionTable *Extended Confusion Table Plot*

---

## Description

Plots an extended confusion table.

## Usage

```
## S3 method for class 'ConfusionTable'
plot(
  x = NULL,
  classNames = NULL,
  main = "extended confusion table",
  xlab = "original class labels",
  ylab = "predictions",
  cascLab = "cascade",
  otherLab = "other classes",
  digits = 3,
  ignore = 0,
  casc.colors = c("#f5f5f5", "#8c510a"),
  other.colors = c("#f5f5f5", "#01665e"),
  colSep = "#b2182b",
  las = 1,
  color.key = TRUE,
  cex = 1,
  cex.lab = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A ConfusionTable object as it is returned by `confusion.table`-function. |
| `classNames` | Vector of the original class names. If not given the class number is used instead. |
| `main` | See `plot`. |
| `xlab` | A title for the x axis (see `plot`). |
| `ylab` | A title for the y axis (see `plot`). |
| `cascLab` | Character string used as header for the cascade part of the extended confusion table. |
| `otherLab` | Character string used as header for the other class part of the extended confusion table. |
| `digits` | Integer indicating the number of decimal places to be used (see `round`). |
| `ignore` | A numeric value between 0 and 1. All confusion and purity values below this number are not written as string into the corresponding element. |
| `casc.colors` | A 2-element vector of the color for the minimal and maximal class-wise sensitivity of the first class. The color palette is calcuated by an interpolation between the 2 given colors. |
| `other.colors` | A 2-element vector of the color for the minimal and maximal class-wise sensitivity of the second class. The color palette is calcuated by an interpolation between the 2 given colors. |
| `colSep` | Color, which is used for the vertical line separating the cascade classes and the other classes. |
| `las` | See `par`. |
| `color.key` | Specifies whether a color key is drawn (TRUE) or not (FALSE). |
| `cex` | See `par`. |
| `cex.lab` | See `par`. |
| `...` | Further arguments passed from other methods. |

## Value

No return value, called to generate the confusion table plot.

## See Also

`confusion.table`, `plot.Subcascades`, `plot.Conf`

## Examples

```
library(TunePareto)
data(esl)
data <- esl$data
labels <- esl$labels
foldList <- generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap <- gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
conf.table <- confusion.table(genMap,cascade='0>1>3>4',other.classes = 'all')

plot(conf.table)
```

---

`plot.Subcascades` *Heatmap of a Subcascades Object*

---

## Description

Plots a heatmap, that shows the performance of cascades of a Subcascades object.

## Usage

```
## S3 method for class 'Subcascades'
plot(
  x = NULL,
  class.sort = "",
  row.sort = "sens",
  thresh = 0,
  main = "subcascades",
  xlab = "classes",
  ylab = "cascades",
  digits = 3,
  ignore = 0,
  casc.colors = c("#f5f5f5", "#01665e"),
  na.color = c("#f5f5f5"),
  color.key = TRUE,
  las = 1,
  cex = 1,
  cex.lab = 1,
  srt = 30,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A Subcascades object as it is returned by `subcascades`-function. |
| `class.sort` | The classes can be sorted either according to the first cascade in the subcacades object (”), or based on the class frequency ('max'). |
| `row.sort` | The cascade can be sorted either based on the maximal-minimal class-wise sensitivity ('sens') or according the class frequency. |
| `thresh` | A numeric value between 0 and 1 that is the minimal sensitivity of the Subcascades object used as lower bound for the color scaling. |
| `main` | See `plot`. |
| `xlab` | A title for the x axis (see `plot`). |
| `ylab` | A title for the y axis (see `plot`). |
| `digits` | Integer indicating the number of decimal places to be used (see `round`). |
| `ignore` | A numeric value between 0 and 1. All confusion and purity values below this number are not written as string into the corresponding element. |
| `casc.colors` | A 2-element vector of the color for the minimal and maximal class-wise sensitivity. The color palette is calcuated by an interpolation between the 2 given colors. |
| `na.color` | Color, which is used for indicating the empty elements (if the given class is not part of the cascade). |
| `color.key` | Specifies whether a color key is drawn (TRUE) or not (FALSE). |
| `las` | See `par`. |
| `cex` | See `par`. |
| `cex.lab` | See `par`. |
| `srt` | Angle used to rotate the strings of the x-axis and y-axis labels (see `par`). |
| `...` | Further arguments passed from other methods. |

## Details

This function plots a heatmap with the cascades of the Subcascades object in the rows and all classes present in any of the cascades in the columns. The colors indicate whether a given class is present in the corresponding cascade and with which sensitivity.

## Value

No return value, called to generate the heatmap plot of the subcascades Object.

## See Also

`subcascades`, `plot.Conf`, `plot.ConfusionTable`

## Examples

```
library(TunePareto)
data(esl)
data <- esl$data
labels <- esl$labels
foldList <- generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap <- gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades <- subcascades(genMap,thresh=0.7,size=c(3,4))

plot(subcascades,thresh=0.7,row.sort='max')
```

---

subcascades *Subcascades Evaluation*

---

## Description

Subcascades returns all cascades found within the data or evaluates a set of specific cascades.

## Usage

```
subcascades(
  predictionMap = NULL,
  sets = NULL,
  thresh = 0,
  size = NA,
  numSol = 1000
)
```

## Arguments

| | |
|---|---|
| predictionMap | A PredictionMap object as it is returned by gen.predictionMap-function. It is made up of two elements(pred and meta). The rownames of the pred-matrix (e.g. [0vs1]) show the classes of the binary base classifier. The elements are the prediction result of a specific training. The rows that correspond to base classifiers that would separate the same class consists of -1. Those rows are not used within the analysis. The meta information connects the values in the pred-matrix to a specific fold, run and contains the original label. |
| sets | Contains the set used for filtering. It is either a list of numberic vectors, a numeric vector, or a vector of characters representing a cascade of the following format '1>2>4'. |

| | |
|---|---|
| `thresh` | A numeric value between 0 and 1. The minimal sensitivity threshold used to filter the returned cascades. Only cascades that pass this threshold are returned. If 0 is used the returned cascades are filtered for >0 and otherwise >= thresh. For low thresholds the calculation lasts longer. |
| `size` | A numeric value that defines the size of the cascades that should be returned. The smallest size is 2 and the largest the maximal number of classes of the current dataset. If size is NA (the default setting), all cascades from 2 to the maximal number of classes are evaluated. |
| `numSol` | The maximum number of cascades that should pass the first sensitivity bound and are further evaluated. |

## Details

This function can either be used to evaluate the performance of a specific cascade, a set of cascades or to filter out the set of cascades of a specific size and passing a given threshold. If the sets-variable is given no size can be set.

## Value

A Subcascades object comprising the evaluated cascades and their performances. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade.

## See Also

`plot.Subcascades`, `summarySubcascades`

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(),  kernel='linear')

# use default parameter settings
# -> returns cascades of all lengths that show a minimal classwise sensitivity >0.
subcascades = subcascades(genMap)
# change the threshold
# -> returns cascades of all lengths that show a minimal classwise sensitivity >=0.6.
subcascades = subcascades(genMap, thresh=0.6)
# search only for cascades of length 2 and 4
# -> returns cascades of length 2 and 4 that show a minimal classwise sensitivity >=0.6.
subcascades = subcascades(genMap, thresh=0.6, size=c(2,4))
# evaluates the performance of the cascade '0>1>2>3>4'.
subcascades = subcascades(genMap, sets = c('0>1>2>3>4'))
```

---

`summary.Subcascades` *Summary Subcascades Characteristics*

---

## Description

Generates a general overview of the characteristics of the Subcascades object.

## Usage

```
## S3 method for class 'Subcascades'
summary(object = NULL, digits = 3, ...)
```

## Arguments

| | |
|---|---|
| `object` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| `digits` | Integer defining the number of decimal places as it is used in the `round`-function. |
| `...` | Further arguments passed from other methods. |

## Details

This function gives a general overview of characteristics of the Subcascades object, like number of cascades or maximal cascade length.

## Value

An invisible Subcascades object as it is returned by `subcascades`-function. The function is called to print a summary of the Subcascades object to console.

## See Also

`subcascades`, `summarySubcascades`, `summaryGroupwise`, `summaryClasses`

### Examples

```
library(TunePareto)
data(esl)
data <- esl$data
labels <- esl$labels
foldList <- generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap <- gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades <- subcascades(genMap,thresh=0.7,numSol=10000)

summary(subcascades)
```

---

## `summaryClasses` *Occurrence of Classes by Size*

---

### Description

`summaryClasses` returns the occurrence of classes by size

### Usage

```
summaryClasses(subcascades = NULL)
```

### Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |

### Details

This function gives an overview of the classes of the Subcascades object. For each length in the Subcascades object the occurence of classes is given.

### Value

A matrix summarizing the overview characteristics of the Subcascades object.

## See Also

subcascades, summarySubcascades, summaryGroupwise

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.7)

summaryClasses(subcascades)
```

---

## summaryGroupwise *Overview Class Groups*

---

### Description

summaryGroupwise returns a summarizing overview. For each length the number of permutations consisting of the same set of classes is given.

### Usage

```
summaryGroupwise(subcascades = NULL, maxCl = 50)
```

### Arguments

| | |
|---|---|
| subcascades | A Subcascades object as it is returned by subcascades-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |
| maxCl | An integer defining the lower bound for the maximal number of classes. Has only to be set if the analyzed dataset has more than 50 classes. |

### Details

This function gives an overview of the subgroup characteristics of the Subcascades object. The number of permutations per size is given. A permutation means that the corresponding cascades contain the same classes but with different order.

### Value

A matrix summarizing the overview characteristics of the Groupwise object.

### See Also

`subcascades`, `summarySubcascades`, `summaryClasses`

### Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
foldList = generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap = gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')
# generate Subcascades object
subcascades = subcascades(genMap,thresh=0.7)

summaryGroupwise(subcascades)
```

## `summarySubcascades` *Overview Subcascades*

### Description

`summarySubcascades` returns a summarizing overview. For each length the number of cascades and the minimal class-wise sensitivity is given.

### Usage

```
summarySubcascades(subcascades = NULL)
```

## Arguments

| | |
|---|---|
| `subcascades` | A Subcascades object as it is returned by `subcascades`-function. The Subcascades object is made up of a list of matrices. Each matrix comprises the evaluation results of cascades of a specific length and is sorted row-wise according to the achieved minimal classwise sensitivities of the cascades (decreasing). The rownames show the class order by a character string of type '1>2>3' and the entries the sensitivity for each position of the cascade. |

## Details

This function gives an overview of the Subcascades object. For each length in the Subcascades object the number of cascades of that length, as well as their minimal classwise sensitivity is given.

## Value

A matrix summarizing the overview characteristics of the Subcascades object. For each size (rows) the number of cascades within the Subcascades object (number) and the minimal classwise sensitivity (min.class.sens) are given.

## See Also

`plot.Subcascades`, `subcascades`, `summaryGroupwise`, `summaryClasses`

## Examples

```
library(TunePareto)
data(esl)
data <- esl$data
labels <- esl$labels
foldList <- generateCVRuns(labels  = labels,
                          ntimes      = 2,
                          nfold       = 2,
                          leaveOneOut = FALSE,
                          stratified  = TRUE)
genMap <- gen.predictionMap(data, labels, foldList = foldList,
classifier = tunePareto.svm(), kernel='linear')

# use default parameter settings
subcascades <- subcascades(genMap,thresh=0.7)
summarySubcascades(subcascades)
```

---

`tunePareto.occ` *Ordinal Classifier Cascade Tune Pareto Object*

---

## Description

TunePareto wrapper for the ordinal classifier cascade.

## Usage

```
tunePareto.occ(base.classifier)
```

## Arguments

| | |
|---|---|
| `base.classifier` | A predefined TuneParetoClassifier object used as binary classifier for the ordinal classifier cascade. There exist five classifier types that can be used: tunePareto.knn(), tunePareto.svm(), tunePareto.tree(), tunePareto.randomForest(), tunePareto.NaiveBayes(). For more information about these classifier functions please refer to the corresponding help page of like `TunePareto::tunePareto.knn`. |

## Details

The "tunePareto.occ" encapsulates the classifier of an ordinal classifier cascade. Additionally, to the parameters of the corresponding base classifier the "class.order" parameter can be provided.

## Value

Returns an object of class TuneParetoClassifier (see: `TunePareto::tuneParetoClassifier`). This can be passed to the function trainTuneParetoClassifier (see: `TunePareto::trainTuneParetoClassifier`).

## Examples

```
library(TunePareto)
data(esl)
data = esl$data
labels = esl$labels
# train classifier
model <- trainTuneParetoClassifier(
         classifier  = tunePareto.occ( base.classifier = tunePareto.svm()),
         trainData   = data,
         trainLabels = labels,
         class.order = as.character(c(4,3,1,0)),
         kernel      = "linear",
         cost        = 1)

# predict labels
prediction <- predict(object = model, newdata = data)
```

# Index