Package ‘optimalFlow’

March 7, 2024

Type Package
Title optimalFlow
Version 1.14.0
Author Hristo Inouzhe <hristo.inouzhe@gmail.com>
Maintainer Hristo Inouzhe <hristo.inouzhe@gmail.com>
Description Optimal-transport techniques applied to supervised flow cytometry gating.
License Artistic-2.0
Encoding UTF-8
LazyData true
Depends dplyr, optimalFlowData, rlang (>= 0.4.0)
Imports transport, parallel, Rfast, robustbase, dbscan, randomForest, foreach, graphics, doParallel, stats, flowMeans, rgl, ellipse
Suggests knitr, BiocStyle, rmarkdown, magick
VignetteBuilder knitr
biocViews Software, FlowCytometry, Technology
RoxygenNote 7.1.0
git_url https://git.bioconductor.org/packages/optimalFlow
git_branch RELEASE_3_18
git_last_commit 9d31fbcc
Repository Bioconductor 3.18
Date/Publication 2024-03-06

R topics documented:

costWasserMatchingEllipse .................................................. 2
cytoPlot ............................................................................. 3
cytoPlot3d .......................................................................... 4
cytoPlotDatabase ............................................................... 5
Description

Calculates a similarity distance based on the 2-Wasserstein distance between mixtures of multivariate normal distributions.

Usage

```r
costWasserMatchingEllipse(
  test.cytometry,
  training.cytometries,
  equal.weights = FALSE
)
```

Arguments

test.cytometry  A clustering represented as a list of clusters. Each cluster is a list with elements mean, cov, weight and type.

training.cytometries  A list of clusterings with the same format as test.cytometry.

equal.weights  If True, weights assigned to every cluster in a partition are uniform (1/number of clusters) when calculating the similarity distance. If False, weights assigned to clusters are the proportions of points in every cluster compared to the total amount of points in the partition.
**Value**

A vector representing the similarity distance between test.cytometry and the elements in training.cytometries.

**References**


**Examples**

```r
partition1 <- list(list(mean = c(1, 1), cov = diag(1, 2), weight = 0.5, type = '1'),
                  list(mean = c(-1, -1), cov = diag(1, 2), weight = 0.5, type = '2'))
partition2 <- list(list(list(mean = c(1, -1), cov = diag(1, 2),
                        weight = 0.5, type = '1'),
                  list(mean = c(-1, 1), cov = diag(1, 2), weight = 0.5, type = '2')))

costWasserMatchingEllipse(partition1, partition2)
```

**Description**

A plot wrapper for cytometries as a mixture of multivariate normals as used in optimalFlowTemplates.

**Usage**

```r
cytoPlot(
  cytometry.as.mixture,
  dimensions = c(1, 2),
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL
)
```

**Arguments**

- `cytometry.as.mixture`:
  A list, where each element contains the parameters of a component of the mixture as a list with entries: mean, cov, weight and type.

- `dimensions`:
  A vector containing the two variables on which to perform the projection.

- `xlim`:
  The x limits (x1, x2) of the plot. Note that x1 > x2 is allowed and leads to a ‘reversed axis’. The default value, NULL, indicates that the range of the finite values to be plotted should be used.

- `ylim`:
  The y limits of the plot.
cytoPlot3d

definition
A rgl::plot3d wrapper for cytometries as a mixture of multivariate normals as used in optimalFlowTemplates.

Usage

cytoPlot3d(
    cytometry.as.mixture,
    dimensions = c(1, 2),
    xlim = NULL,
    ylim = NULL,
    zlim = NULL,
    xlab = NULL,
    ylab = NULL,
    zlab = NULL
)

Arguments

cytometry.as.mixture
A list, where each element contains the parameters of a component of the mixture as a list with entries: mean, cov, weight and type.
dimensions
A vector containing the three variables on which to perform the projection.
**Description**

A plot wrapper for a database (list) of cytometries as a mixture of multivariate normals as used in `optimalFlowTemplates`.

**Usage**

```r
cytoPlotDatabase(
  database = your_database
  dimensions = c(1, 2)
  xlim = c(0, 8000)
  ylim = c(0, 8000)
  xlab = ""
  ylab = ""
  colour = TRUE
)
```

---

**Value**

A three dimensional plot of ellipsoids containing the 95

**Examples**

```r
database <- buildDatabase(
  dataset_names = paste0('Cytometry', c(2:5, 7:9, 12:17, 19, 21)),
  population_ids = c('Monocytes', 'CD4+CD8-', 'Mature SIg Kappa', 'TCRgd-'))
templates.optimalFlow <- optimalFlowTemplates(
  database = database, templates.number = 5, cl.paral = 1
)
# # To execute requires an actual monitor since it uses rgl.
# cytoPlot3d(templates.optimalFlow$templates[[3]], dimensions = c(4, 3, 9), xlim = c(0, 8000), ylim = c(0, 8000), zlim = c(0, 8000), xlab = "", ylab = "", zlab = "")
```
Arguments

database.cytometries.as.mixtures
A list where each component is a mixture distribution. That is, each component
is a list, where each element contains the parameters of a component of the
mixture as a list with entries: mean, cov, weight and type.
dimensions A vector containing the two variables on which to perform the projection.
xlim the x limits (x1, x2) of the plot. Note that x1 > x2 is allowed and leads to a
‘reversed axis’. The default value, NULL, indicates that the range of the finite
values to be plotted should be used.
ylim the y limits of the plot.
xlab a label for the x axis, defaults to a description of x.
ylab a label for the y axis, defaults to a description of y.
colour If TRUE plots elements of a mixture distribution in different colours. If FALSE
plots them in black.

Value
A two dimensional plot of ellipses containing the 95

Examples

database <- buildDatabase(
  dataset_names = paste0('Cytometry', c(2:5, 7:9, 12:17, 19, 21)),
  population_ids = c('Monocytes', 'CD4+CD8-', 'Mature Sig Kappa', 'TCRgd-'))
templates.optimalFlow <-
  optimalFlowTemplates(
    database = database, templates.number = 5, cl.paral = 1
  )
cytoPlotDatabase3d(templates.optimalFlow$database.elliptical[which(templates.optimalFlow$clustering == 3)], dimensions = c(1, 2, 3), xlim = c(0, 8000), ylim = c(0, 8000))

Description
A plot3d wrapper for a database (list) of cytometries as a mixture of multivariate normals as used in optimalFlowTemplates.

Usage

cytoPlotDatabase3d(
  database.cytometries.as.mixtures,
  dimensions = c(1, 2, 3),
  xlim = c(0, 8000),
  ylim = c(0, 8000),
cytoPlotDatabase3d

zlim = c(0, 8000),
xlab = "",
ylab = "",
zlab = "",
colour = TRUE
)

Arguments

database.cytometries.as.mixtures
A list where each component is a mixture distribution. That is, each component is a list, where each element contains the parameters of a component of the mixture as a list with entries: mean, cov, weight and type.
dimensions
A vector containing the two variables on which to perform the projection.
xlim
the x limits (x1, x2) of the plot. Note that x1 > x2 is allowed and leads to a 'reversed axis'. The default value, NULL, indicates that the range of the finite values to be plotted should be used.
ylim
the y limits of the plot.
zlim
the z limits of the plot.
xlab
a label for the x axis, defaults to a description of x.
ylab
a label for the y axis, defaults to a description of y.
zlab
a label for the z axis, defaults to a description of z.
colour
If TRUE plots elements of a mixture distribution in different colours. If FALSE plots them in black.

Value
A three dimensional plot of ellipsoids containing the 95

Examples

database <- buildDatabase(
    dataset_names = paste0('Cytometry', c(2:5, 7:9, 12:17, 19, 21)),
    population_ids = c('Monocytes', 'CD4+CD8-', 'Mature Sig Kappa', 'TCRgd-'))
templates.optimalFlow <- optimalFlowTemplates(
    database = database, templates.number = 5, cl.paral = 1
)
# # To execute requires an actual monitor since it uses rgl.
# cytoPlotDatabase3d(templates.optimalFlow$database.elliptical[which(templates.optimalFlow$clustering == 3)], dim
estimationCellBarycenter

Description
Estimates a Wasserstein barycenter for a cluster type using a collection of partitions.

Usage
estimationCellBarycenter(cell, cytometries)

Arguments
- cell: Name of the cluster of interest.
- cytometries: List of clusterings.

Value
A list representing the (1-)barycenter:
- mean: Mean of the barycenter.
- cov: Covariance of the barycenter.
- weight: Weight associated to the barycenter.
- type: Type of the cluster.

Examples
partition1 <- list(list(mean = c(1, 1), cov = diag(1, 2), weight = 0.5, type = '1'),
                   list(mean = c(-1, -1), cov = diag(1, 2), weight = 0.5, type = '2'))
partition2 <- list(list(mean = c(1, -1), cov = diag(1, 2), weight = 0.5, type = '1'),
                   list(mean = c(-1, 1), cov = diag(1, 2), weight = 0.5, type = '2'))
cytometries <- list(partition1, partition2)
estimationCellBarycenter('1', cytometries)

estimCovCellGeneral

Description
Estimation of mean and covariance for a label in a partition.

Usage
estimCovCellGeneral(cell, cytometry, labels, type = "standard", alpha = 0.85)
**Arguments**

- **cell**
  Labell of the cluster of interest.

- **cytometry**
  Data of the partition, without labels.

- **labels**
  Labels of the partition.

- **type**
  How to estimate covariance matrices of a cluster. 'standard' is for using cov(), while 'robust' is for using robustbase::covMcd.

- **alpha**
  Only when type = 'robust'. Indicates the value of alpha in robustbase::covMcd.

**Value**

A list containing:

- **mean**
  Mean of the cluster.

- **cov**
  Covariance of the cluster.

- **weight**
  Weight associated to the cluster.

- **type**
  Type of the cluster.

**Examples**

```r
estimCovCellGeneral('Basophils', Cytometry1[,1:10], Cytometry1[,11])
```

---

**f1Score**

**Description**

Calculates the F1 score for each group in a partition.

**Usage**

`f1Score(clustering, cytometry, noise.cells)`

**Arguments**

- **clustering**
  The labels of the new classification.

- **cytometry**
  Data of the clustering, where the last variable contains the original labels.

- **noise.cells**
  An array of labels to be considered as noise.

**Value**

A matrix where the first row is the F1 score, the second row is the Precision and the third row is the Recall.
References

Examples
f1Score(dplyr::pull(Cytometry3[c(sample(1:250,250),251:(dim(Cytometry3)[1])),],11), Cytometry3, noise.types)

f1ScoreVoting

Description
Calculates the F1 score for each group in a partition, when provided with a fuzzy classification.

Usage
f1ScoreVoting(voting, clustering, cytometry, nivel_sup, noise.cells)

Arguments
voting            A list where each entry is a vote on the respective label.
clustering        Labels of the partition.
cytometry         Data of the clustering, where the last variable contains the original labels.
nivel_sup         level of tolerance for assigning a hard clustering. Should be greater or equal than 1. Class A is assigned if class A > nivel_sup * Class B.
noise.cells       An array of labels to be considered as noise.

Value
A matrix where the first row is the F1 score, the second row is the Precision and the third row is the Recall.

Examples
# We construct a simple database selecting only some of the Cytometries and some cell types for simplicity and for a
database <- buildDatabase(
  dataset_names = paste0('Cytometry', c(2:5, 7:9, 12:17, 19, 21)),
  population_ids = c('Monocytes', 'CD4+CD8-', 'Mature Sig Kappa', 'TCRgd-'))

templates.optimalFlow <- optimalFlowTemplates(database = database, templates.number = 5, cl.paral = 1)

classification.optimalFlow <- optimalFlowClassification(as.data.frame(Cytometry1)[
  which(match(Cytometry1$'Population ID (name)'[,11), Cytometry3, noise.types)
Description

Label transfer between a test partition and a training set of partitions.

Usage

```
labelTransfer(
  training.cytometry,  
  test.cytometry,  
  test.partition,  
  equal.weights = FALSE
)
```

Arguments

- **training.cytometry**
  List of partitions, where each partition is a dataframe where the last column contains the labels of the partition.

- **test.cytometry**
  Test data, a dataframe without labels.

- **test.partition**
  Labels of a partition of the test data.

- **equal.weights**
  If True, weights assigned to every cluster in a partition are uniform (1/number of clusters) when calculating the similarity distance. If False, weights assigned to clusters are the proportions of points in every cluster compared to the total amount of points in the partition.

Value

A fuzzy re-labeling consistent of a transportation plan.

Examples

```
data.example <- data.frame(v1 = c(rnorm(50, 2, 1), rnorm(50, -2, 1)),  
                            v2 = c(rnorm(50, 2, 1), rnorm(50, -2, 1)),  
                            id = c(rep(0, 50), rep(1, 50)))  
test.labels <- c(rep('a', 50), rep('b', 50))  
labelTransfer(data.example, data.example[, 1:2], test.labels)
```
Description

Label transfer between a test partition and a training partitions viewed as a mixture of gaussians.

Usage

\[
\text{labelTransferEllipse}(\text{i, test.cytometry.ellipses, training.cytometries.barycenter, equal.weights = FALSE})
\]

Arguments

- **i**: A dummy variable, should be any integral. Ment for use with lapply.
- **test.cytometry.ellipses**: A test clustering viewed as a mixture of multivariate normal distributions.
- **training.cytometries.barycenter**: A training partition viewed as a mixture of multivariate normal distributions.
- **equal.weights**: If True, weights assigned to every cluster in a partion are uniform (1/number of clusters) when calculating the similarity distance. If False, weights assigned to clusters are the proportions of points in every cluster compared to the total amount of points in the partition.

Value

A fuzzy relabeling consistent of a transportation plan.

References


Examples

\[
\begin{aligned}
\text{partition1} &\leftarrow \text{list(list(mean = c(1, 1), cov = diag(1, 2), weight = 0.5, type = '1'), list(mean = c(-1, -1), cov = diag(1, 2), weight = 0.5, type = '2'))} \\
\text{partition2} &\leftarrow \text{list(list(mean = c(1, 1), cov = diag(1, 2), weight = 0.5, type = 'a'), list(mean = c(-1, -1), cov = diag(1, 2), weight = 0.5, type = 'b'))} \\
\text{labelTransferEllipse}(1, \text{partition2, partition1})
\end{aligned}
\]
optimalFlowClassification

optimalFlowClassification

Description

Performs a supervised classification of input data when a database and a partition of the database are provided.

Usage

optimalFlowClassification(
    X,
    database, 
    templates, 
    consensus.method = "pooling", 
    cov.estimation = "standard", 
    alpha.cov = 0.85, 
    initial.method = "supervised", 
    max.clusters = NA, 
    alpha.tclust = 0, 
    restr.factor.tclust = 1000, 
    classif.method = "qda", 
    qda.bar = TRUE, 
    cost.function = "points", 
    cl.paral = 1, 
    equal.weights.voting = TRUE, 
    equal.weights.template = TRUE
)

Arguments

X          Datasample to be classified.
database   A list where each entry is a partition (clustering) represented as dataframe, of the same dimensions, where the last variable represents the labels of the partition.
templates  List of the consensus clusterings for every group in the partition of the database obtained by optimalFlowTemplates
consensus.method The consensus.method value that was used in optimalFlowTemplates.
cov.estimation How to estimate covariance matrices in each cluster of a partition. "standard" is for using cov(), while "robust" is for using robustbase::covMcd.
alpha.cov    Only when cov.estimation = "robust". Indicates the value of alpha in robustbase::covMcd.
initial.method Indicates how to obtain a partition of X. Takes values in c("supervised", "unsupervised"). Supervised uses tclust initialized by templates. Unsupervised uses flowMeans.
max.clusters The maximum numbers of clusters for flowMeans. Only when initial.method = unsupervized.

alpha.tclust Level of trimming allowed fo tclust. Only when initial.method = supervised.

restr.factor.tclust Fixes the restr.fact parameter in tclust. Only when initial.method = supervised.

classif.method Indicates what type of supervised learning we want to do. Takes values on c("matching", "qda", "random forest").

qda.bar Only if classif.method = "qda". If True then the appropriate consensus clustering (template, prototype) is used for learning. If False, the closest partition in the appropriate group is used.

cost.function Only if classif.method = "matching". Indicates the cost function, distance between clusters, to be used for label matching.

c1.paral Number of cores to be used in parallel procedures.

equal.weights.voting only when classif.method = "qda" and qda.bar =F, or when classif.method = "random forest". Indicates the weights structure when looking for the most similar partition in a group.

equal.weights.template If True, weights assigned to every cluster in a partion are uniform (1/number of clusters). If False, weights assigned to clusters are the proportions of points in every cluster compared to the total amount of points in the partition.

Value

A list formed by:

cluster Labels assigned to the input data.

clusterings A list that contains the initial unsupervised or semi-supervised clusterings of the cytometry of interest. Can have as much entries as the number of templates in the semi-supervised case (initial.method = "supervised"), or only one entry in the case of initial.method = "unsupervised". Each entry is a list where the most relevant argument for the clusterings is cluster.

assigned.template.index Label of the group for which the template is closer to the data. When classical qda or random forest are used for classification there is a second argument indicating the index of the cytometry in the cluster used for learning.

cluster.vote Only when classif.method = "matching" or when consensus.method in c("hierarchical", "k-barycenter"). Vote on the type of every label in the partition of the data. In essence, cluster + cluster.vote return a fuzzy clustering of the data of interest.

References

optimalFlowTemplates

Examples

```r
# # We construct a simple database selecting only some of the Cytometries and some cell types for simplicity and for a better visualisation.
database <- buildDatabase(
  dataset_names = paste0('Cytometry', c(2:5, 7:9, 12:17, 19, 21)),
  population_ids = c('Monocytes', 'CD4+CD8-', 'Mature SIg Kappa', 'TCRgd-'))

# # To select the appropriate number of templates, via hierarchical tree, in an interactive fashion and produce a clustering.
models.optimalFlow <- optimalFlowTemplates(database = database)

classification.optimalFlow <- optimalFlowClassification(Cytometry1[
  which(match(Cytometry1$`Population ID (name)` , c("Monocytes", "CD4+CD8-", "Mature SIg Kappa", "TCRgd-"), nomatch = 0) > 0), 1:10], database, models.optimalFlow, cl.paral = 1)

scoreF1.optimalFlow <- optimalFlow::f1Score(classification.optimalFlow$cluster,
  Cytometry1[which(match(Cytometry1$`Population ID (name)` , c("Monocytes", "CD4+CD8-", "Mature SIg Kappa", "TCRgd-"), nomatch = 0) > 0),], noise.types)
```

optimalFlowTemplates

Description

Returns a partition of the input clusterings with a respective consensus clustering for every group.

Usage

```r
optimalFlowTemplates(
  database,
  database.names = NULL,
  cov.estimation = "standard",
  alpha.cov = 0.85,
  equal.weights.template = TRUE,
  hclust.method = "complete",
  trimm.template = FALSE,
  templates.number = NA,
  minPts = 2,
  eps = 1,
  consensus.method = "pooling",
  barycenters.number = NA,
  bar.repetitions = 40,
  alpha.bar = 0.05,
  bar.ini.method = "plus-plus",
  consensus.minPts = 3,
  cl.paral = 1
)
```
Arguments

**database**
A list where each entry is a partition (clustering) represented as dataframe, of the same dimensions, where the last variable represents the labels of the partition.

**database.names**
Names of the elements in the database.

**cov.estimation**
How to estimate covariance matrices in each cluster of a partition. 'standard' is for using cov(), while 'robust' is for using robustbase::covMcd.

**alpha.cov**
Only when cov.estimation = 'robust'. Indicates the value of alpha in robustbase::covMcd.

**equal.weights.template**
If True, weights assigned to every cluster in a partition are uniform (1/number of clusters). If False, weights assigned to clusters are the proportions of points in every cluster compared to the total amount of points in the partition.

**hclust.method**
Indicates what kind of hierarchical clustering to do with the similarity distances matrix of the partitions. Takes values in c('complete', 'single', 'average', 'hdbscan', 'dbscan').

**trimm.template**
Logical value. Indicates if it is allowed to not take into account some of the entries of database. Default is False.

**templates.number**
Only if hclust.method in c('complete', 'single', 'average'). Indicates the number of clusters to use with cutree. If set to NA (default), plots the hierarchical tree and asks the user to introduce an appropriate number of clusters.

**minPts**
Only if hclust.method in c('hdbscan', 'dbscan'). Indicates the value of argument minPts in dbscan::dbscan and dbscan::hdbscan.

**eps**
Only if hclust.method = 'dbscan'. Indicates the value of eps in dbscan::dbscan.

**consensus.method**
Sets the way of doing consensus clustering when clusters are viewed as Multivariate Distributions. Can take values in c('pooling', 'k-barycenter', 'hierarchical'). See details.

**barycenters.number**
Only if consensus.method = 'k-barycenter'. Sets the number, k, of barycenters when using k-barycenters.

**bar.repetitions**
Only if consensus.method = 'k-barycenter'. How many times to repeat the k-barycenters procedure. Equivalent to nstart in kmeans.

**alpha.bar**
Only if consensus.method = 'k-barycenter'. The level of trimming allowed during the k-barycenters procedure.

**bar.ini.method**
Only if consensus.method = 'k-barycenter'. Takes values in c('rnd', 'plus-plus'). See details.

**consensus.minPts**
Only if consensus.method = 'hierarchical'. The value of argument minPts for dbscan::hdbscan.

**cl.paral**
Number of cores to be used in parallel procedures.
Value

A list containing:

- **templates** A list representing the consensus clusterings for every group in the partition of the database. Each element of the list is a template partition. Hence it is a list itself, containing the cell types in the prototype, where each element has components: mean, cov, weight and type.

- **clustering** Clustering of the input partitions.

- **database.elliptical** A list containing each cytometry in the database viewed as a mixture distribution. Each element of the list is a cytometry viewed as a mixture. Hence it is a list itself, containing the cell types in the cytometry, where each element has components: mean, cov, weight and type.

References


Examples

```r
# We construct a simple database selecting only some of the Cytometries and some cell types for simplicity and for a better visualization.
database <- buildDatabase(
    dataset_names = paste0('Cytometry', c(2:5, 7:9, 12:17, 19, 21)),
    population_ids = c('Monocytes', 'CD4+CD8-', 'Mature SIG Kappa', 'TCRgd-'))

# To select the appropriate number of templates, via hierarchical tree, in an interactive fashion and produce a clustering
templates.optimalFlow <- optimalFlowTemplates(database = database)

templates.optimalFlow <- optimalFlowTemplates(database = database, templates.number = 5,
    cl.paral = 1)
```

Description

Gives quadratic discriminant scores to the points in data for a multivariate normal.

Usage

```
qdaClassification(normal, data)
```

Arguments

- **normal** A list with arguments mean, covariance and weight.
- **data** Data frame or matrix on which to perform qda.
tclustWithInitialization

Value

A score for each point.

Examples

data.qda = cbind(rnorm(50), rnorm(50))
exp(qdaClassification(list(mean = c(0,0), cov = diag(1,2), weight = 1), data.qda))

Description

A wrapper for the function tclust_H.

Usage

tclustWithInitialization(
  initialization, cytometry, i.sol.type = "points", trimming = 0.05, restr.fact = 1000
)

Arguments

initialization Initial solution for parameters provided by the user. Can be a matrix of data containing observations anc cluster assignations or can be a list spesifying a multivariate mixture of gaussians.
cytometry A matrix or data.frame of dimension n x p, containing the observations (row-wise).
i.sol.type Type of initial solutions in c("points", "barycenters"). "points" refers to a classified data matrix, while "barycenters" to a multivariate mixture.
trimming The proportion of observations to be trimmed.
restr.fact The constant restr.fact >= 1 constrains the allowed differences among group scatters. Larger values imply larger differences of group scatters, a value of 1 specifies the strongest restriction.
Value

A list with entries:

---

**cluster** A numerical vector of size n containing the cluster assignment for each observation. Cluster names are integer numbers from 1 to k, 0 indicates trimmed observations.

**n_clus** Number of clusters actually found.

**obj** The value of the objective function of the best (returned) solution.

Examples

```r
x <- rbind(matrix(rnorm(100), ncol = 2), matrix(rnorm(100) + 2, ncol = 2),
            matrix(rnorm(100) + 4, ncol = 2))
## robust cluster obtention from a sample x asking for 3 clusters,
## trimming level 0.05 and constrain level 12
k <- 3; alpha <- 0.05; restr.fact <- 12
output = tclust_H(x = x, k = k, alpha = alpha, nstart = 50, iter.max = 20,
                   restr = 'eigen', restr.fact = restr.fact, sol_ini_p = FALSE, sol_ini = NA,
                   equal.weights = FALSE, trace = 0, zero.tol = 1e-16)
## cluster assigment
output2 <- tclustWithInitialization(data.frame(x, output$cluster), x, 'points', 0.05, 10)
```

---

Description

A wrapper for the internal function tclust_. Performs robust non spherical clustering, tclust, where initial solutions are allowed.

Usage

```r
tclust_H(
  x,
  k = 3,
  alpha = 0.05,
  nstart = 50,
  iter.max = 20,
  restr = "eigen",
  restr.fact = 12,
  sol_ini_p = FALSE,
  sol_ini = NA,
  equal.weights = FALSE,
  trace = 0,
  zero.tol = 1e-16
)
```
Arguments

x  A matrix or data.frame of dimension n x p, containing the observations (rowwise).
k  The number of clusters initially searched for.
alpha  The proportion of observations to be trimmed.
nstart  The number of random initializations to be performed. Only when sol_ini_p = FALSE.
iter.max  The maximum number of concentration steps to be performed. The concentration steps are stopped, whenever two consecutive steps lead to the same data partition.
restr  The type of restriction to be applied on the cluster scatter matrices. Valid values are "eigen" (default).
restr.fact  The constant restr.fact >= 1 constrains the allowed differences among group scatters. Larger values imply larger differences of group scatters, a value of 1 specifies the strongest restriction.
sol_ini_p  Initial solution for parameters provided by the user TRUE/FALSE, if TRUE is stored in sol_ini.
sol_ini  Initial solution for parameters provided by the user.
equal.weights  A logical value, specifying whether equal cluster weights (TRUE) or not (FALSE) shall be considered in the concentration and assignment steps.
trace  Defines the tracing level, which is set to 0 by default. Tracing level 2 gives additional information on the iteratively decreasing objective function's value.
zero.tol  The zero tolerance used. By default set to 1e-16.

Details

This iterative algorithm initializes k clusters randomly and performs "concentration steps" in order to improve the current cluster assignment. The number of maximum concentration steps to be performed is given by iter.max. For approximately obtaining the global optimum, the system is initialized nstart times and concentration steps are performed until convergence or iter.max is reached. When processing more complex data sets higher values of nstart and iter.max have to be specified (obviously implying extra computation time). However, if more then half of the iterations would not converge, a warning message is issued, indicating that nstart has to be increased.

The parameter restr defines the cluster's shape restrictions, which are applied on all clusters during each iteration. Options "eigen"/"deter" restrict the ratio between the maximum and minimum eigenvalue/determinant of all cluster's covariance structures to parameter restr.fact. Setting restr.fact to 1, yields the strongest restriction, forcing all eigenvalues/determinants to be equal and so the method looks for similarly scattered (respectively spherical) clusters. Option "sigma" is a simpler restriction, which averages the covariance structures during each iteration (weighted by cluster sizes) in order to get similar (equal) cluster scatters.

Value

A list with values:
trimmedKBarycenter

- **centers** A matrix of size p x k containing the centers (column-wise) of each cluster.
- **cov** An array of size p x p x k containing the covariance matrices of each cluster.
- **cluster** A numerical vector of size n containing the cluster assignment for each observation. Cluster names are integer numbers from 1 to k, 0 indicates trimmed observations.
- **par** A list, containing the parameters the algorithm has been called with (x, if not suppressed by store.x = FALSE, k, alpha, restr.fact, nstart, KStep, and equal.weights).
- **weights** A numerical vector of length k, containing the weights of each cluster.
- **obj** The value of the objective function of the best (returned) solution.

**References**


**Examples**

```r
x <- rbind(matrix(rnorm(100), ncol = 2),
           matrix(rnorm(100) + 2, ncol = 2),
           matrix(rnorm(100) + 4, ncol = 2))
## robust cluster obtention from a sample x asking for 3 clusters,
## trimming level 0.05 and constrain level 12
k <- 3; alpha <- 0.05; restr.fact <- 12
output <- tclust_H(x = x, k = k, alpha = alpha, nstart = 50, iter.max = 20,
                    restr = "eigen", restr.fact = restr.fact, sol_ini_p = FALSE, sol_ini = NA,
                    equal.weights = FALSE, trace = 0, zero.tol = 1e-16)
## cluster assignment
output$cluster
plot(x, col = output$cluster)
```

**Description**

Calculates a 2-Wasserstein k-barycenter of a list of multivariate normal distributions.

**Usage**

`trimmedKBarycenter(k, alpha0, type.ini = "rnd", reps.list)`

**Arguments**

- **k** Number k of elements in the k-barycenter.
- **alpha0** Level of trimming.
- **type.ini** of initialization in c(rnd', 'plus-plus'). 'rnd' makes the common random initialization while 'plus-plus' initializes in a similar fashion to k-means++.
- **reps.list** List of multivariate normals for which the trimmed k-barycenter should be performed.
Value

A list with values:

- **variacion_wasser** A double giving the Waserstein variation.
- **baricentro** A list of k elements, each of which is a member of the k-barycenter. Each element is a normal distribution characterized by a mean and a covariance.
- **cluster** The assignment of the original entries to each member of the k-barycenter.

Examples

```r
normals <- list(list(mean = c(1, 1), cov = diag(2, 2)),
                list(mean = c(1, 1), cov = diag(1, 2)),
                list(mean = c(3, 3), cov = diag(1, 2)))
trimmedKBarycenter(2, 0, 'rnd', normals)
```

---

**Description**

A wrapper for doing either labelTransfer or labelTransferEllipse.

**Usage**

```r
voteLabelTransfer(
  type = "points",
  test.partition,
  test.cytometry,
  test.partition.ellipse,
  training.cytometries,
  training.cytometries.barycenter,
  test = 1,
  op.syst,
  cl.paral = 1,
  equal.weights = FALSE
)
```

**Arguments**

- **type** 'points’ indicates use of labelTransfer; 'ellipses’ of labelTransferEllipse.
- **test.partition** Only when type = 'points’. Labels of a partition of the test data.
- **test.cytometry** Only when type = 'points’. Test data, a dataframe without labels.
- **test.partition.ellipse** Only when type = 'ellipses’. A test clustering viewed as a mixture of multivariate normal distributions.
training.cytometries
Only when type = 'points'. List of partitions, where each partition is a dataframe where the last column contains the labels of the partition.

training.cytometries.barycenter
Only when type = 'ellipses'. A training partition viewed as a mixture of multivariate normal distributions.

test
Only when type = 'ellipses'. A dummy variable, should be any integral. Ment for use with lapply.

op.syst
Type of system, takes values in c('unix', 'windows').

c1.paral
Number of cores to be used in parallel procedures.

equal.weights
If True, weights assigned to every cluster in a partion are uniform (1/number of clusters) when calculating the similarity distance. If False, weights assigned to clusters are the proportions of points in every cluster compared to the total amount of points in the partition.

Value
A list containing:

final.vote A list for the votes on each cell.

complete.vote A more complete list for the votes on each cell.

Examples

data.example <- data.frame(v1 = c(rnorm(50, 2, 1), rnorm(50, -2, 1)),
v2 = c(rnorm(50, 2, 1), rnorm(50, -2, 1)), id = c(rep(0, 50), rep(1, 50)))
test.labels <- c(rep('a', 50), rep('b', 50))
voteLabelTransfer(test.partition = test.labels, test.cytometry = data.example[, 1:2],
training.cytometries = list(data.example), op.syst = .Platform$OS.type)$final.vote[[1]]

w2dist

Description
The 2-Wasserstein distance between two multivariate normal distributions.

Usage
w2dist(P, Q)

Arguments

P A multivariate normal distribution given as a list with arguments mean and cov.
Q A multivariate normal distribution given as a list with arguments mean and cov.
Value

A double giving the 2-Wasserstein distance between the two distributions.

Examples

P <- list(mean = c(1, 1), cov = diag(1, 2))
Q <- list(mean = c(0, 0), cov = 1.1*diag(1, 2))
w2dist(P, Q)

Description

Calculates the similarity distance between elements j and i of a list of partitions.

Usage

wasserCostFunction(j, i, cytometries, equal.weights = FALSE)

Arguments

j  An entry of the list of partitions.
i  An entry of the list of partitions.
cytometries  The list of partitions.
equal.weights  If True, weights assigned to every cluster in a partion are uniform (1/number of clusters) when calculating the similarity distance. If False, weights assigned to clusters are the proportions of points in every cluster compared to the total amount of points in the partition.

Value

A double giving the value of the similarity distance.

Examples

# We construct a simple database selecting only some of the Cytometries and some cell types for simplicity and for a better visualisation.
database <- buildDatabase(
  dataset_names = paste0('Cytometry', c(2:5, 7:9, 12:17, 19, 21)),
  population_ids = c('Monocytes', 'CD4+CD8-', 'Mature Sig Kappa', 'TCRgd-'))

templates.optimalFlow <- optimalFlowTemplates(database = database, templates.number = 5,
                                             cl.paral = 1)
print(wasserCostFunction(1, 2, list(templates.optimalFlow$database.elliptical[[1]],
                                  templates.optimalFlow$database.elliptical[[2]])))
Index

costWasserMatchingEllipse, 2
cytoPlot, 3
cytoPlot3d, 4
cytoPlotDatabase, 5
cytoPlotDatabase3d, 6
estimationCellBarycenter, 8
estimCovCellGeneral, 8
f1Score, 9
f1ScoreVoting, 10
labelTransfer, 11
labelTransferEllipse, 12
optimalFlowClassification, 13
optimalFlowTemplates, 15
qdaClassification, 17
tclust_H, 19
tclustWithInitialization, 18
trimmedKBarycenter, 21
voteLabelTransfer, 22
w2dist, 23
wasserCostFunction, 24