Package 'UNCLES'

October 12, 2022

Type Package

Title Unification of Clustering Results from Multiple Datasets using External Specifications

Version 2.0

Date 2016-06-28

Author Basel Abu-Jamous

Maintainer Basel Abu-Jamous <baselabujamous@gmail.com>

Imports pdist, kohonen, class

Depends R (>= 3.1.0)

Description Consensus clustering by the unification of clustering results from multiple datasets using external specifications.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-29 10:37:45

R topics documented:

`UNCLES-package` *Unification of Clustering Results from Multiple Datasets using External Specifications*

Description

UNCLES package

Author(s)

Basel Abu-Jamous

`binarise` *Binarise*

Description

Binarise

Usage

```
binarise(U, K, technique = "DTB", parameter = 0.0)
```

Arguments

`U`

`K`

`technique`

`parameter`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (U, K, technique, parameter)
{
    return(NULL)
  }
```

`closestToSquareFactors` *closestToSquareFactors*

Description

Finds the two factors of (n), closest to its square root

Usage

```
closestToSquareFactors(n)
```

Arguments

```
n
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (n)
{
    return(NULL)
  }
```

clustDist *clustDist*

Description

Find the distances between clusters

Usage

```
clustDist(U1, U2, X, criterion)
```

Arguments

U1

U2

X

criterion

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(U1, U2, X, criterion) {
    return(NULL)
}
```

clusterDataset *Cluster Dataset*

Description

Cluster a single dataset

Usage

```
clusterDataset(X, K, D = 0, methods = list(kmeansKA))
```

Arguments

X

K

D

methods

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X, K, D = 0, methods = c("HC-Ward", "kmeans", "SOMs"))
{
    return(NULL)
  }
```

---

clustVec2partMat *clustVec2partMat*

Description

Converts a clustering result vector to a partition matrix

Usage

```
clustVec2partMat(C)
```

Arguments

```
C
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(C) {
    return(NULL)
}
```

---

factors *factors*

Description

Find the factors of a number. If (primesonly) was true, it gives prime factors only. Otherwise, it gives all factors

Usage

```
factors(n, primeonly = FALSE)
```

Arguments

```
n
```

```
primeonly
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (n, primeonly = FALSE)
{
    return(NULL)
  }
```

`fixnans` *Fixnans*

Description

Fixing the not a number (NaN) entries in a dataset

Usage

```
fixnans(X, type = "spline")
```

Arguments

```
X
```

```
type
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X, type = "spline")
{
    return(NULL)
  }
```

fuzzystretch *fuzzystretch*

Description

Stretches the fuzzy values in each row of the matrix X such the one stays one, zeros stays zero, x0 stays x0, values between zero and x0 decreased, and values between x0 and one increased.

If a vector was given for x0 then it should have elements with the same number of rows in X. If a single value of x0 was given then it is used for all of the rows of X.

The default value of x0 is the mean of the non-zero elements of the corresponding row, we recommend using this value because it preserve the criterion of (sum of fuzzy values for a single row is unity) to a good level.

Usage

```
fuzzystretch(X, x0 = -1)
```

Arguments

X

x0

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X, x0 = -1)
{
    return(NULL)
  }
```

generateCoPaM *Generate CoPaM*

Description

Generate a Consensus Partition Matrix (CoPaM) from partitions

Usage

```
generateCoPaM(U, relabel_technique, w, X, distCriterion, K, GDM)
```

Arguments

U

relabel_technique

w

X

distCriterion

K

GDM

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (U, relabel_technique = "minmin", w = numeric(), X = numeric(),
    distCriterion = "direct_euc", K = 0, GDM = numeric())
{
    return(NULL)
  }
```

HC *HC*

Description

Perform hierarchical clustering

Usage

```
HC(X, K, distancemetric = "euclidean", method = "ward.D2")
```

Arguments

X

K

distancemetric	Any distance method which can be used by the function "dist" Default: "euclidean"
method	The hierarchical clustering method / algorithm. Acceptable values are: "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC) Default: "ward.D2"

isempty *isempty*

Description

Checks if the input matrix, vector, or list X is empty

Usage

```
isempty(X)
```

Arguments

X

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X)
{
    return(NULL)
  }
```

isnullorempty *isnullorempty*

Description

Checks if the input matrix, vector, or list X is NULL or empty

Usage

```
isnullorempty(X)
```

Arguments

X

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X)
{
    return(NULL)
  }
```

---

isValidBPM *isValidBPM*

Description

Checking if a matrix is a valid binary partition matrix (BPM)

Usage

```
isValidBPM(U)
```

Arguments

```
U
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (U)
{
    return(NULL)
  }
```

KA *KA*

Description

Perform Kaufman's initialisation for k-means clustering

Usage

```
KA(X,K,distancemetric)
```

Arguments

X

K

distancemetric

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X,K,distancemetric)
{
    return(NULL)
  }
```

kmeansKA *kmeansKA*

Description

Perform kmeans clustering using Kaufman's (KA) initialisation

Usage

```
kmeansKA(X, K, distancemetric, iter.max, nstart, algorithm, trace)
```

Arguments

X

K

distancemetric

iter.max

nstart

algorithm

trace

---

mnplots *M-N Plots*

Description

Generate M-N plots and select clusters based on them

Usage

```
mnplots(unclesResult, MCs = 10, corner = c(0, 1),
    removedtype = 'abs', removedval = 1,
   Vmse = numeric(), mseCache = numeric(), doplot = FALSE, subplotdim = numeric(),
    subplotind = 1:MCs, minimiseDistance = TRUE)
```

Arguments

unclesResult	The result of the "uncles" function as it is, or a similarly structred list as explained here. This argument is a list which must include two named elements at least: - unclesResult$X: A list of the datasets as matrices or data frames. If a single dataset is provided, it may be provided as it is rather than as a list containing a single matrix or data frame. - unclesResult$B: This can simply be a list of partitions, or can be a multi-dimensional list array of partitions. Any of the partitions, for example unclesResult$B[[i]], is a binary partition of M rows representing M genes, and K columns representing K clusters. The element unclesResult$B[[i]][j,k] is the binary membership (either 1 or 0) of the (j)th gene in the (k)th cluster as per the (i)th partition. The partitions can have different numbers of clusters (K values) represented by their columns, but they all have to have the same numbers of genes, represented by rows. Moreover, the gene represented by the (j)th row in one of the partitions has to be the same gene represented by the (j)th row in all of the other partitions. In other words, the rows of the partitions must be aligned. If this argument is passed as the output of the "uncles" function, it will be a 4D list array of binary partitions with the dimensions of:

(T)x(NBP1)x(NBP2)x(NKs).

where (T) is the number of the CoPaM final trials; (NBP1) is the number of the different values of the parameter of the binarisation technique if the UNCLES type is "A", and is the number of the different values of the parameter of the positive binarisation technique if the UNCLES type is "B"; (NBP2) is 1 if the UNCLES type is "A", and is the number of the different values of the parameter of the negative binarisation technique if UNCLES type is "B"; (NKs) is the number of the different numbers of clusters (K values). For example: if 5 trials of the final CoPaM were considered, UNCLES type "A" was used with a DTB binarisation technique whose parameter delta ranges from 0.0 to 1.0 with steps of 0.1, and 4 different K values were considered (e.g. K = 4, 8, 12, and 16), then the dimensions of unclesResult$B will be (5x11x1x4).

Other optional named elements of the argument unclesResult include:

- unclesResult$GDM: Gene-dataset logical matrix of M rows representing M genes and L columns representing L datasets. A value of 1 in an element of this matrix indicates that the corresponding gene is found in the corresponding dataset, i.e. it is represented by some probe(s) in that dataset.

Default: All ones (all considered genes are found in all of the datasets).

- unclesResult$params$type: The type of UNCLES, 'A' or 'B'. Default: 'A'.

- unclesResult$params$setsP: For UNCLES type 'B', these are the datasets considered in the positive set of datasets. See the description of the argument "setsP" of the function "uncles".

- unclesResult$params$setsN: For UNCLES type 'B', these are the datasets considered in the negative set of datasets. See the description of the argument "setsN" of the function "uncles".

- unclesResult$params$wsets: For L datasets, this is a vector of L numeric values representing the relative weights of the datasets. The vector does not have to be normalised as it will be normalised within the "uncles" function. Valid examples for 5 datasets include:

wsets = c(0.2, 0.2, 0.2, 0.2, 0.2)

wsets = rep(1, 5)

wsets = c(4, 4, 4, 4, 4)

wsets = c(1, 2, 2, 0, 1)

wsets = c(0.2, 0.3, 0, 0.4, 0.4)

Note that the first three examples result in the same weighting, which is to treat all datasets equally. If the weight of a dataset was set to zero, this implies excluding it of the analysis.

Default: numeric() # which will be read as equal weights for all datasets.

`MCs` The number of clusters to be selected by the M-N scatter plots technique. This is also the number of iterations, as in each iteration one cluster is selected.

Default: 10.

`corner` The coordinates of point at the unity-normalised M-N plots which is considered as the reference point from which the distance is measured for the points of all of the clusters. Better clusters are those which are closer to this corner. Its default is the top-left corner of the plot with the coordinates of (0.0, 1.0). As the horizontal axis represents the dispersion within the cluster and the vertical axis

represents the size of the cluster, clusters closer to that top-left corner minimise dispersion while maximize their size.

If the reference was moved a bit towards the right on the horizontal axis (e.g. to become at (0.2, 1.0)), wider clusters will be selected. While if it was moved towards the left (e.g. to become at (-0.2, 1.0)), tighter clusters will be selected.

Default: c(0.0, 1.0).

`removedtype` This is either 'perc' or 'abs'. When a cluster is selected as the best cluster (closest to the "corner" argument), how do we identify the other clusters which overlap with it?

Read the description of the argument "removedval" below for details.

Default: 'abs'

`removedval` A numeric value indicating the minimum amount of overlap between the cluster selected as the best cluster in the current iteration and the other clusters for these other clusters to be removed before the following iteration.

If "removedtype" is 'perc', "removedval" represents the percentage of the overlap out of the smaller cluster between the two clusters being compared. "removedval" in this case should be in the range 0.0 to 1.0. For example, if "removedval" is 0.25, the overlap between the two clusters has to be at least 25% of the smaller cluster of the two to consider it a significant overlap.

If "removedtype" is 'abs', "removedval" represents the minimum number of genes in the overlap to consider it as a significant overlap. "removedval" in this case should be an integer greater than zero.

Default: 1. As the default of "removedtype" is 'abs', this means that if a single gene was found in the overlap, the overlap is considered significant.

`Vmse`

`mseCache`

`doplot` If TRUE, the function plots the M-N plots in addition to providing the calculated results in the output. If FALSE, it just calculates the results and provides them without plotting.

`subplotdim`

`subplotind`

`minimiseDistance`

Examples

```
# This is the simplist way to apply UNCLES and MN plots.
# Just pass the datasets to the "uncles" function and then pass
# the UNCLES result to the "mnplots" function.
# Both functions will use default values for all other arguments.
#
# Define three random gene expression datasets for 1000 genes.
# The number of samples in the datasets are 6, 4, and 9, respectively.
#
# X = list()
# X[[1]] = matrix(rnorm(6000), 1000, 6)
```

```
# X[[2]] = matrix(rnorm(4000), 1000, 4)
# X[[3]] = matrix(rnorm(9000), 1000, 9)
#
# unclesResult <- uncles(X)
# mnResult <- mnplots(unclesResult)
#
# The clusters will be available in the form of a partition matrix in the variable:
# mnResult$B;
```

`mseclusters` *MSE Clusters*

Description

Calculate the MSE values for clusters.

Usage

```
mseclusters(X, B, normalise = TRUE)
```

Arguments

`X`

`B`

`normalise`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

`normaliseMatrix` *Normalise Matrix*

Description

Normalise a given matrix

Usage

```
normaliseMatrix(X, type)
```

Arguments

`X`

`type`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (X, type)
{
    return(NULL)
  }
```

---

partMat2clustVec *partMat2clustVec*

Description

Converts a partition matrix to a clustering result vector

Usage

```
partMat2clustVec(U, skipValidity=FALSE)
```

Arguments

U

skipValidity

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(U, skipValidity=FALSE) {
    return(NULL)
}
```

permutations *permutations*

Description

Finds all permutations of numbers 1 to n

Usage

```
permutations(n)
```

Arguments

n

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (n)
{
    return(NULL)
  }
```

relabelClusts *Relabel Clusters*

Description

Relabel clusters

Usage

```
relabelClusts(ref, input, technique = "minmin", X = numeric(),
                  distCriterion = "direct_euc")
```

Arguments

ref

input

technique

X

distCriterion

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (Ref, Input, technique = "minmin", X = numeric(), distCriterion = "direct_euc")
{
    return(NULL)
  }
```

SOMs *SOMs*

Description

Perform self-organising maps (SOMs) clustering

Usage

```
SOMs(X, K, topo = "hexagonal", rlen = 100, alpha = c(0.05, 0.01),
    n.hood = "circular")
```

Arguments

X

K

topo

rlen

alpha

n.hood

---

uncles *UNCLES*

Description

Perform UNCLES clustering

Usage

```
uncles(X, type = 'A', Ks = c(4, 8, 12, 16),
methods = list(kmeansKA, list(HC, method = "ward.D2"), SOMs),
methodsDetailed = list(), inparams = list(), normalise = 0,
samplesIDs = numeric(), flipSamples = list(), U = list(),
UType = 'PM', Xn = list(), relabel_technique = "minmin",
binarisation_technique = "DTB", binarisation_param = seq(0, 1, 0.1),
setsP = numeric(), setsN = numeric(),
dofuzzystretch = FALSE, wsets = numeric(), wmethods = numeric(),
GDM = numeric(), CoPaMforDatasetTrials = 1, CoPaMfinaltrials = 1)
```

Arguments

X	The datasets to be clustered as a list of matrices or data frames. If a single dataset is to be provided, it can be as a matrix, a data frame, or a list of a single matrix or frame.
type	The type of UNCLES. Either "A" or "B". Default: "A"
Ks	A vector with the K values (numbers of clusters) with each of which UNCLES will be performed to the same datasets. The result of the function will include the results of applying UNCLES to each one of them. Default: c(4, 8, 12, 14)
methods	A list of the individual clustering methods' functions to be employed by UNCLES. Each method can be given as a single function (e.g. kmeansKA) or as a list with the methods' function as its first element followed by named elements representing the arguments to be passed to that clustering function. If the clustering function returns a list of multiple elements and one of which is the actual clustering result, add a named element to the list with the case-sensitive name "outputVariable" to indicate the name of the variable or element in the list which includes the clustering result (as a vector of cluster indices or a partition matrix). For example: method1 = kmeansKA method2 = list(kmeans, outputVariable = "cluster", iter.max=100) method3 = list(HC, method = "ward.D2") method4 = list(HC, method = "average") method5 = list(SOMs, topo = "rectangular") methods = list(method1, method2, method3, method4, method5) Default: list(kmeansKA, list(HC, method = "ward.D2"), SOMs)
methodsDetailed	If you wish to apply different methods to different datasets, use this argument. For L datasets, this is a list of L lists. Each list represents the methods to be applied to its corresponding dataset and has the format of the argument "methods" above. If this argument was not provided, or of it was empty, the methods in the argument "methods" will be applied to all datasets. Default: list()
inparams	As this uncles method includes in its output a variable "params" to store the parameters used in it, you may provide input parameters here which will be

passed to the output as they are after the addition of the uncles parameters. This is useful if you wish to add your own parameters.

Example:

params = list()

params$author = "Basel Abu-Jamous"

params$studytitle = "Analysis of gene expression"

result = uncles(..., inparams = params)

the output "result$params" here will have all uncles parameters in addition to "author" and "studytitle".

`normalise` For L datasets, this is a list of L normalisation values. Each element is a single number or a vector of numbers representing the normalisation techniques to be applied to the corresponding dataset in order. For example, if three datasets were included, "normalise" can be:

list(6, c(3, 2), 6)

which applies normalisation (6) to the first and the third datasets, and applies the normalisation techniques (3) and (2), in order, to the second dataset.

If a single value or a single vector was provided, it is applied to all datasets.

Refer to the help of the "normaliseMatrix" function for details on normalisation techniques' codes.

Default: 0 (no normalisation)

`samplesIDs` If some datasets include replicates that need summarisation, this must be provided. For L datasets, this is a list of L vectors of integers. Each vector has to be equal in length to the number of samples in corresponding dataset. Each integer in one of these vectors represents the index of the group of replicates to which the corresponding sample belongs. The value (0) is provided to indicate that the corresponding sample should not be included.

For example, consider this samplesIDs list for 3 different datasets:

samplesIDs = list(c(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 0, 0, 0, 0),

1:10,

c(3, 2, 1, 3, 2, 1, 2, 0))

The first vector in these three vectors within samplesIDs matches the real dataset GSE22552, which has 16 samples. The first 12 of them represent 3 replicates for each of the four stages of erythropoiesis (CFU-E, Pro-E, Int-E, and Late-E). The last 4 samples of the 16 are unsorted samples (not to be included).

The second vector represents a dataset with 10 independant samples with no replicates.

The third vector represents a dataset with 3 groups of samples (1, 2, and 3) with replicates that are not ordered in the original dataset as desired in the output.

In any case, the replicates are summarised by taking their median value, and are ordered in the processed and normalised datasets starting from the group numbered as (1), followed by (2), and so on.

Default: numeric() (i.e. no replicates are combined, and provided datasets are clustered as they are)

`flipSamples` This is provided if some samples in the datasets are replicates in opposite directions (e.g. dyes in two-colour arrays were flipped for the condition and the

same experimental condition), and therefore some of them needs to be "flipped" by taking their reciprocals or opposite sign before they can be summarised by median.

For L datasets, this is a list of L numeric vectors. This is an example for 4 datasets:

flipSamples = list(numeric(), numeric(), c(0, 0, 2, 0, 2, 0), numeric())

In this example, the first, second, and fourth datasets need no flipping of their samples. On the other hand, two samples of the third dataset, indicated with the flipping code (2), need to be negated by toggeling their sign before they are considered in summarisation as per the "samplesIDs" argument described above and indeed before clustering.

Flipping codes are:

0: no flipping 1: flipping by taking the reciprocal (1/x) 2: flipping by negating (-x)

Default: list() (i.e. no flipping for any dataset)

`U` If you already have the individual clustering results, provide them here. For L datasets, this should be a list matrix of L rows and as many columns as the used K values. Each element of this matrix will be a list of partition matrices that all use the same K value but may have been generated using different individual clustering methods or runs.

For example:

U[[i,1]] is a list of partitions of the (i)th dataset that have the same number of clusters (K). U[[i,1]][[1]] might be a partition produced by k-means clustering, U[[i,1]][[2]] might be a partition produced by self- organising maps (SOMs) clustering, and so on.

For the same (i)th dataset, U[[i,2]] is a nother list of partitions which have a similar K value to each other but different from U[[i,1]].

The format of each partition U[[i,j]][[l]] depends on the value of the argument "UType". See details in the description of that argument below. The default of UType is "PM".

If (U) is provided, the arguments "methods" and "methodsDetailed" will be ignored.

Default: list()

`UType` This is the type of the partitions the argument "U" if provided. This can be "PM" for partition matrices or "IDX" for cluster index vectors.

If UType is "PM", a partition U[[i,j]][[l]] should be a partition matrix of K rows representing clusters and M columns representing the clustered objects (e.g. genes in gene clustering). Each value U[[i,j]][l]][l,m] is the membership value of the (m)th gene in the (k)th cluster, and ranges from 0.0 (does not belong) to 1.0 (fully belongs).

If UType is "IDX", a partition U[[i,j]][[l]] should be a vector of M integer elements (for M genes). Each value U[[i,j]][[l]][m] is an integer that represents the index of the cluster to which the (m)th gene belongs. Therefore, if the total number of clusters is (K), this value would range from 1 to K. However, if the value is zero, it indicates that this gene does not belong to any cluster.

Default: "PM"

`Xn` Normalised datasets. This has the same format of the argument "X", and if provided, normalisation using the "normalise" argument will be ignored.

Default: list()

`relabel_technique`

The relabelling technique to be used for the relabelling step. This can have one of these values:

- "brute": Brute force relabelling. This is not practical for K > 8.
- "minmin_strict": minmin relabelling
- "minmax_strict": minmax relabelling
- "minmin" (DEFAULT): if (K > 8), minmin relabelling is applied, otherwise brute force is applied.
- "minmax": if (K > 8), minmax relabelling is applied, otherwise brute force is applied.

`binarisation_technique`

This is one of the six binarisation techniques described in (Abu-Jamous et al., PLOS ONE, 2013):

- "MVB": maximum value binarisation
- "IB": intersection binarisation
- "UB": union binarisation
- "TB": top binarisation
- "VTB": value threshold binarisation
- "DTB" (DEFAULT): difference threshold binarisation

TB, VTB, and DTB require the next argument "binarisation_param".

`binarisation_param`

If the "binarisation_technique" argument is TB, VTB, or DTB, this argument is considered. It is the tuning parameter that is associated with those techniques as described in (Abu-Jamous et al., PLOS ONE, 2013).

Default: seq(0, 1, 0.1)

`setsP` For UNCLES type "B", this is a vector of integers representing the indices of the positive datasets in X. If the number of datasets is L, every element of setsP should be between 1 and L, inclusively.

For UNCLES type "A", a concatenation of both setsP and setsN is formed to represent the datasets to be considered. In other words, if the concatenation c(setsP, setsN) does not include all of the integers from 1 to L, the missing indices represent the indices of the datasets to be ignored in the UNCLES "A" analysis.

For example, if 8 datasets were provided in X (L = 8), and:

type = "A"

setsP = c(1, 2, 3, 6)

setsN = c(4, 5, 8)

This means that UNCLES A will be applied over the datasets 1, 2, 3, 6, 4, 5, and 8, while the dataset 7 will be ignored.

The X and Xn members of the result of UNCLES will include 7 datasets only in the order 1, 2, 3, 6, 4, 5, and then 8.

Default (if Type = "A"): 1:L

Default (if Type = "B"): 1:(ceiling of L/2)

setsN

For UNCLES type "B", this is a vector of integers representing the indices of the negative datasets in X. If the number of datasets is L, every element of setsP should be between 1 and L, inclusively.

For UNCLES type "A", see the description of the "setsP" argument.

Default (if Type = "A"): numeric()

Default (if Type = "B"): 1:(floor of L/2).

dofuzzystretch

When multiple clustering methods are applied to multiple datasets, the partitions resulting from applying mutiple methods to the same dataset are first combined to obtain an intermediate consensus partition matrix (CoPaM) per dataset, then these intermediate CoPaMs are combined to produce the final CoPaM.

If "dofuzzystretch" is set to TRUE, the intermediate CoPaMs are "fuzzy stretched" before they are combined to produce the final CoPaM. Fuzzy stretching is to push their fuzzy values closer to 0.0 and 1.0, i.e. to make them less fuzzy and closer to binary. This makes the effect of the differences amongst the datasets on the final result stronger than the effect of the differences amongst the clustering methods. See the description of the "fuzzystrech" function for details on the equations used to perform fuzzy stretching.

Default: FALSE

wsets

For L datasets, this is a vector of L numeric values representing the relative weights of the datasets. The vector does not have to be normalised as it will be normalised within the "uncles" function. Valid examples for 5 datasets include:

wsets = c(0.2, 0.2, 0.2, 0.2, 0.2)

wsets = rep(1, 5)

wsets = c(4, 4, 4, 4, 4)

wsets = c(1, 2, 2, 0, 1)

wsets = c(0.2, 0.3, 0, 0.4, 0.4)

Note that the first three examples result in the same weighting, which is to treat all datasets equally. If the weight of a dataset was set to zero, this implies excluding it of the analysis.

Default: numeric() # which will be read as equal weights for all datasets.

wmethods

Similar format to "wsets" but as a vector with the same length as the number of methods used, as it represents weights of the different clustering methods.

GDM

Gene-Dataset Matrix (GDM) is provided if not all of the datasets have probesets/entries for the same set of genes. GDM is an MxL matrix where M is the total number of genes from all datasets and L is the number of the datasets. GDM[m,l] is 1 if the (m)th gene is included in the (l)th dataset, and is 0 if it does not.

For example, if there are 6 genes in total (M = 6) and 3 datasets (L = 3), a possible GDM can be:

GDM =

1 1 1

1 0 1

1 1 1

1 1 1

0 1 1

1 1 1

This means that each one of the first and the second datasets has 5 genes only, while the third has all of the six genes. It is important that the rows of the datasets in the argument X are in the same order as the order in the GDM matrix.

Default: numeric() # which will consider that all datasets X[[1]] to X[[L]] have the same number of rows representing genes, and in the same order.

`CoPaMforDatasetTrials`

Number of different CoPaMs generated for each dataset by combining the partitions generated for that dataset.

This is used because the combining process takes one of the partitions to be combined as the reference and then applies relabelling and merging for the rest of them one by one. Practice shows that different order of partitions in this merging may produce different results. Therefore, generating more than one CoPaM for the same dataset using different random permutations, which are combined to produce the final CoPaM afterwards, may produce more robust results.

Default: 1.

`CoPaMfinaltrials`

Number of different final CoPaMs generated.

UNCLES first combines the different partitions generated for any single dataset into a single CoPaM per dataset per K value, or as many as the argument "CoPaMforDatasetTrials" states if it was provided. Then, these per-set CoPaMs are combined to produce the final CoPaM. For the same reason for which the argument "CoPaMforDatasetTrials" may be provided, that is, because different orders of combining of the partitions or per-set CoPaMs into a CoPaM may produce different results, this argument also is provided.

In the final output, the variable "params$CoPaMs" for type A or the variables "params$CoPaMsP" and "params$CoPaMsN" for type B, are list matrices with "CoPaMfinaltrials" rows and as many columns as the number of different K values, i.e. the number of elements in the argument "Ks". For example:

result = uncles(...) result$params$CoPaMs[[i,j]] is a CoPaM (numeric partition matrix) produced by the (i)th trial of combining the per-set CoPaMs of all datasets at the (j)th K value.

Also, the first dimension of the four dimensions of the output "B" is this number of trials as well.

Indeed, larger values of this argument enlarges the output, while larger values of the previous argument "CoPaMforDatasetTrials" does not, as all trials of per-set CoPaMs are eventually combined into the same output fuzzy CoPaM(s) or binary B(s).

Default: 1

Examples

```
# This is the simplist way to apply UNCLES and MN plots.
# Just pass the datasets to the "uncles" function and then pass
# the UNCLES result to the "mnplots" function.
```

```
# Both functions will use default values for all other arguments.
#
# Define three random gene expression datasets for 1000 genes.
# The number of samples in the datasets are 6, 4, and 9, respectively.
#
# X = list()
# X[[1]] = matrix(rnorm(6000), 1000, 6)
# X[[2]] = matrix(rnorm(4000), 1000, 4)
# X[[3]] = matrix(rnorm(9000), 1000, 9)
#
# unclesResult <- uncles(X)
# mnResult <- mnplots(unclesResult)
#
# The clusters will be available in the form of a partition matrix in the variable:
# mnResult$B;
```

Index