Package 'CancerSubtypesPrognosis'

May 19, 2021

Type Package

Title Cancer subtypes and prognosis based on multiple genomic data sets

Version 1.1.0

Date 2021-02-19

Author Xiaomei Li<xiaomei.li@mymail.unisa.edu.au>
Taosheng Xu<taosheng.x@gmail.com>
Thuc Le<Thuc.Le@unisa.edu.au>

Maintainer Xiaomei Li<xiaomei.li@mymail.unisa.edu.au>

Depends R (>= 3.6), sigclust, NMF, Biobase, stringr, genefu, scales

Imports SNFtool, iCluster, cluster, impute, limma, CIMLR,IntNMF, ConsensusClusterPlus, grDevices, survival, PINSPlus, breastCancerMAINZ, survcomp, iC10, doParallel, foreach

Encoding UTF-8

LazyData TRUE

Description Breast cancer is an extremely complex disease. Accurate prognosis and identification of subtypes of breast cancer are important steps towards effective and personalised treatments. To this end, many computational methods have been developed to take advantage of a large amount of available transcriptomic data for breast cancer subtype discovery and prognosis. However, it raises challenges as to how and when to use these methods in practice. We create an R/Bioconductor package, CancerSubtypesPrognosis, to include all the 34 methods to facilitate the reproducibility of the methods and streamline the evaluation. We expect this work can provide biomedical researchers a practical guide to select the appropriate methods and a one-stop software tool to apply the methods to their breast cancer data. We also hope the work can help with the development of new computational methods for breast cancer subtyping and prognosis.

License GPL (>= 2)

Suggests BiocGenerics, RUnit, knitr, RTCGA.mRNA, RTCGA.clinical

VignetteBuilder knitr

biocViews Clustering, Software, Visualization, GeneExpression

URL https://github.com/XiaomeiLi1/CancerSubtypesPrognosis

BugReports https://github.com/XiaomeiLi1/CancerSubtypesPrognosis/issues

RoxygenNote 7.1.1

NeedsCompilation no

R topics documented:

binarize — *binarize a vector by the mediate value*

Description

binarize a vector by the mediate value

Usage

```
binarize(x = NULL, na.rm = TRUE)
```

Arguments

`x`	A numeric vector.
`na.rm`	A logical value indicating whether NA values should be stripped before the computation proceeds.

Value

The binarized vector

CancerPrognosis_LncRNAData — *Evaluate Cancer Prognosis of Long non-coding RNA expression data to compute the risk scores*

Description

Cancer Prognosis to compute the risk scores for long non-coding RNA data based on the 6 methods: "HOTAIR", "MALAT1", "DSCAM-AS1", "lncRNA5","lncRNA6","lncRNA12".

Usage

```
CancerPrognosis_LncRNAData(data, platform = "custom", methods)
```

Arguments

`data`	data to be computed for cancer Prognosis risk scores; either a data matrix or ExpressionSet object. If it is a data matrix, rows= lncRNAs annotated with lncRNA symbols and columns=tems/samples .
`methods`	A set of methods to be performed in the 6 methods:"HOTAIR", "MALAT1", "DSCAM-AS1", "lncRNA5","lncRNA6","lncRNA12".

Value

A dataframe object with rows for samples and columns which represent dataset used and its corresponding methods

References

- **HOTAIR**: Paw lowska E, Szczepanska J, Blasiak J. The Long Noncoding RNA HOTAIR in Breast Cancer: Does Autophagy Play a Role? International journal of molecular sciences. 2017;18(11):2317.
- **MALAT1**: Wang Z, Katsaros D, Biglia N, Shen Y, Fu Y, Loo LW, et al. High expression of long non-coding RNA MALAT1 in breast cancer is associated with poor relapse-free survival. Breast cancer research and treatment. 2018; p. 1-11.
- **DSCAM-AS1**: Niknafs YS, Han S, Ma T, Speers C, Zhang C, Wilder-Romans K, et al. The lncRNA landscape of breast cancer reveals a role for DSCAM-AS1 in breast cancer progression. Nature communications. 2016;7:12791.
- **lncRNA12**: Zhou M, Zhong L, Xu W, Sun Y, Zhang Z, Zhao H, et al. Discovery of potential prognostic long non-coding RNA biomarkers for predicting the risk of tumor recurrence of breast cancer patients. Scientific reports. 2016;6:31038.
- **lncRNA6**: Zhong L, Lou G, Zhou X, Qin Y, Liu L, Jiang W. A six-long non-coding RNAs signature as a potential prognostic marker for survival prediction of ER-positive breast cancer patients. Oncotarget. 2017;8(40):67861.
- **lncRNA5**: Li J, Wang W, Xia P, Wan L, Zhang L, Yu L, et al. Identification of a five-lncRNA signature for predicting the risk of tumor recurrence in breast cancer patients. International journal of cancer. 2018.

Examples

```
data(TCGA500)
methods <- c("HOTAIR", "MALAT1", "DSCAM-AS1", "lncRNA12","lncRNA6","lncRNA5")
res = CancerPrognosis_LncRNAData(data=TCGA500, platform="custom", methods=methods)
```

CancerPrognosis_miRNAData

Evaluate Cancer Prognosis of miRNA expression data to compute the risk scores in 4 methods

Description

Cancer Prognosis to compute the risk scores for miRNA RNA expression data based on existing 4 methods: "hsa-miR-21", "hsa-miR-155", "hsa-miR-210", "miRNA10"

Usage

```
CancerPrognosis_miRNAData(data, methods)
```

Arguments

`data`	data to be computed for Cancer Prognosis risk scores; either a data matrix or ExpressionSet object. If it is a data matrix, rows= miRNAs and columns=terms/samples.
`methods`	A set of methods to be performed in the 4 methods: "hsa-miR-21", "hsa-miR-155", "hsa-miR-210", "miRNA10".

Value

A dataframe object with rows for samples and columns which represent dataset used and its corresponding methods

References

- **hsa-miR-21**: Lee JA, Lee HY, Lee ES, Kim I, Bae JW. Prognostic implications of microRNA-21 overexpression in invasive ductal carcinomas of the breast. Journal of breast cancer. 2011;14(4):269-275. Yan LX, Huang XF, Shao Q, Huang MY, Deng L, Wu QL, et al. MicroRNA miR-21 overexpression in human breast cancer is associated with advanced clinical stage, lymph node metastasis and patient poor prognosis. Rna. 2008;. Markou A, Yousef GM, Stathopoulos E, Georgoulias V, Lianidou E. Prognostic significance of metastasis-related microRNAs in early breast cancer patients with a long follow-up. Clinical chemistry. 2013; p. clinchem-2013.
- **hsa-miR-155**: Gasparini P, Cascione L, Fassan M, Lovat F, Guler G, Balci S, et al. microRNA expression profiling identifies a four microRNA signature as a novel diagnostic and prognostic biomarker in triple negative breast cancers. Oncotarget. 2014;5(5):1174.
- **hsa-miR-210**: Camps C, Buffa FM, Colella S, Moore J, Sotiriou C, Sheldon H, et al. hsa-miR-210 Is induced by hypoxia and is an independent prognostic factor in breast cancer. Clin Cancer Res. 2008;14(5):1340-1348. Wang J, Zhao J, Shi M, Ding Y, Sun H, Yuan F, et al. Elevated expression of miR-210 predicts poor survival of cancer patients: a systematic review and meta-analysis. PLoS One. 2014;9(2):e89223.
- **miRNA10**: Buffa FM, Camps C, Winchester L, Snell CE, Gee HE, Sheldon H, et al. microRNA associated progression pathways and potential therapeutic targets identified by integrated mRNA and microRNA expression profiling in breast cancer. Cancer research. 2011; p. canres-0489.

Examples

```
data(TCGA500)
methods <- c("hsa-miR-21", "hsa-miR-155", "hsa-miR-210", "miRNA10")
res = CancerPrognosis_miRNAData(data=TCGA500, methods=methods)
```

CancerPrognosis_RNAData

Evaluate Cancer Prognosis of coding RNA expression data to compute the risk scores in 13 methods

---

Description

Cancer Prognosis to compute the risk scores for coding RNA expression data based on existing 13 methods: "AURKA", "ESR1", "ERBB2", "GGI", "GENIUS", "Endopredict", "OncotypeDx","TAMR13", "PIK3CAGS", "GENE70", "rorS", "RNAmodel","Ensemble"

Usage

```
CancerPrognosis_RNAData(data, platform = "custom", methods, RNASig = NULL)
```

Arguments

`data` data to be computed for Cancer Prognosis risk scores; either a data matrix or ExpressionSet object. If it is a data matrix, rows= probes/genes annotated with Entrez ID and columns=terms/samples .

`platform` The technical platform for data, "affy" or "agilent" or "custom". Defualt "custom" represnts unknow

`methods` A set of methods to be performed in the 13 methods: "AURKA", "ESR1", "ERBB2", "GGI", "GENIUS", "Endopredict", "OncotypeDx","TAMR13", "PIK3CAGS", "GENE70", "rorS", "RNAmodel","Ensemble", The "Ensemble" method is the average of the five methods, i.e. "GENIUS", "Endopredict", "OncotypeDx", "GENE70", "rorS".

`RNASig` A dataframe represnts the customized signatures provided for the "RNAmodel" method. The columns should include "Gene.symbol","EntrezGene.ID", "weight". Default NULL, the default 30 RNA signatures will be used.

Value

A dataframe object with rows for samples and columns which represent dataset used and its corresponding methods.

Examples

```
## Not run:
library("breastCancerMAINZ")
data("mainz")
methods <- c("AURKA", "ESR1", "ERBB2", "GGI", "GENIUS", "Endopredict", "OncotypeDx",
             "TAMR13", "PIK3CAGS", "GENE70", "rorS", "RNAmodel", "Ensemble")
res = CancerPrognosis_RNAData(data=mainz, platform="custom", methods=methods)

## End(Not run)
```

`CancerSubtypes` *Evaluate Cancer CancerSubtyping methods of mRNA, miRNA or multiomics data based on the running time, the average Silhouette score, and the p-value of Logrank test*

Description

CancerSubtypes to compute the running time, the average Silhouette score, and the p-value of Logrank test for mRNA, miRNA or multiomics data based on existing 11 methods: "PAM50","IntClust","CC","CNMF","iC
CC","WSNF","CIMLR","PINS","NEMO"

Usage

```
CancerSubtypes(
  dn = NULL,
  omics = NULL,
  methods = NULL,
  fileFolder = NULL,
  logFile = NULL
)
```

Arguments

`dn`	datasets to be computed for cancer subtypes; a string vector.
`omics`	The types of data, "mRNA" or "miRNA" or "multiomics".
`methods`	A set of methods to be performed in the 11 methods: "PAM50","IntClust","CC","CNMF", "iCluster","SNF","SNF-CC","WSNF","CIMLR","PINS","NEMO".
`fileFolder`	A file folder name provided for saving results.
`logFile`	A file name provided for saving log information.

Value

a list contains timeTable matrix, silTable matrix, and pvalueTable matrix

Examples

```
## Not run:
dn = c("TCGA","UK","HEL","GSE19783")
methods = c("PAM50","IntClust","CC","CNMF","iCluster","SNF","SNF-CC","WSNF","CIMLR","PINS","NEMO")
omics = "mRNA"
res = CancerSubtypes(dn, omics, methods, fileFolder=omics, logFile = omics)

## End(Not run)
```

`Cindex` *C-index calculation*

Description

Calculating Concordance Indices for the evaluation results of Cancer Prognosis methods

Usage

```
Cindex(data, survival, PValue = FALSE, outputFolder = NULL)
```

Arguments

`data`	A dataframe object with rows for samples and columns represent corresponding methods. A return value from CancerPrognosis_xxx() function
`survival`	A dataframe object which contains variables (columns) representing for survival time and event. The rows are the samples.
`PValue`	if ture, return the object of function concordance.index().
`outputFolder`	(Optional) A desired folder to put the results or "output" by default

Value

This function is used for its side-effect. A plot for overall Concordance Index. For C-Index for each method, please refer in the outputFolder

Examples

```
library("breastCancerMAINZ")
data("mainz")
methods <- c("AURKA", "ESR1", "ERBB2", "GGI", "GENIUS", "Endopredict", "OncotypeDx",
            "TAMR13", "PIK3CAGS", "GENE70", "rorS", "RNAmodel", "Ensemble")
sampleInfo= pData(mainz)
survival=data.frame(time=sampleInfo$t.dmfs,event=sampleInfo$e.dmfs, row.names=sampleInfo$samplename)
## Not run:
res = CancerPrognosis_RNAData(data=mainz, platform="custom", methods=methods)
CIs = Cindex(data=res, survival,outputFolder="./mainz")

## End(Not run)
```

data.checkDistribution

Data check distribution

Description

Data check distribution

Usage

```
data.checkDistribution(Data)
```

Arguments

`Data`	A matrix representing the genomic data such as gene expression data, miRNA expression data. For the matrix, the rows represent the genomic features, and the columns represent the samples.

Value

A plot describes the mean, variance and Median Absolute Deviation (MAD) distribution of features.

Examples

```
data(GeneExp)
data.checkDistribution(GeneExp)
```

`data.imputation` *Data imputation*

Description

Data imputation for features with missing values

Usage

```
data.imputation(Data, fun = "median")
```

Arguments

`Data` A matrix representing the genomic data such as gene expression data, miRNA expression data.
For the matrix, the rows represent the genomic features, and the columns represent the samples.

`fun` A character value representing the imputation type. The optional values are shown below:

- "median". The NAs will be replaced by the median of the existing values of this feature in all samples.
- "mean". The NAs will be replaced by the mean of the existing values of this feature in all samples.
- "microarray". It will apply the "impute" package to impute the missing values. This is a common way to process the missing observation for MicroArray dataset.

Value

The data matrix after imputation (without NAs).

Examples

```
Data=matrix(runif(1000),nrow = 50,ncol = 20)
geneName=paste("Gene", 1:50, sep = " ")
sampleName=paste("Sample", 1:20, sep = " ")
rownames(Data)=geneName
colnames(Data)=sampleName
index=sample(c(1:1000),60)
Data[index]=NA
result=data.imputation(Data,fun="median")
```

`data.normalization` *Data normalization*

Description

Conduct normalization for dataset.

Usage

```
data.normalization(Data, type = "feature_Median", log2 = FALSE)
```

Arguments

`Data`	A matrix representing the genomic data such as gene expression data, miRNA expression data. For the matrix, the rows represent the genomic features, and the columns represent the samples.
`type`	A character value representing the normalization type. The optional values are shown below: • "feature_Median". The default value. Normalize dataset by sweeping the median values of each feature. • "feature_Mean". Normalize dataset by sweeping the mean values of each feature. • "feature_zscore". Conduct z_score normalization for each feature. • "sample_zscore". Conduct z_score normalization for each samples.
`log2`	A logical value. If TRUE, the data is transform as log2(x+1). This is commonly used for RNAseq data.

Value

The normalized data matrix.

Examples

```
data(GeneExp)
result=data.normalization(GeneExp,type="feature_Median",log2=FALSE)
```

`DiffExp.limma` *DiffExp.limma*

Description

Differently Expression Analysis for genomic data. We apply limma package to conduct the analysis.

Usage

```
DiffExp.limma(
  Tumor_Data,
  Normal_Data,
  group = NULL,
  topk = NULL,
  sort.by = "p",
  adjust.method = "BH",
  RNAseq = FALSE
)
```

Arguments

`Tumor_Data` A matrix representing the genomic data of cancer samples such as gene expression data, miRNA expression data.
For the matrix, the rows represent the genomic features, and the columns represent the cancer samples.

`Normal_Data` A matrix representing the genomic data of Normal samples.
For the matrix, the rows represent the genomic features corresponding to the Tumor_Data, and the columns represent the normal samples.

`group` A vector representing the subtype of each tumor sample in the Tumor_Data. The length of group is equal to the column number of Tumor_Data.

`topk` The top number of different expression features that we want to extract in the return result.

`sort.by` This is a parmeter of "topTable() in limma pacakge". "Character string specifying statistic to rank genes by. Possible values for topTable and toptable are "logFC", "AveExpr", "t", "P", "p", "B" or "none". (Permitted synonyms are "M" for "logFC", "A" or "Amean" for "AveExpr", "T" for "t" and "p" for "P".) Possibilities for topTableF are "F" or "none". Possibilities for topTreat are as for topTable except for "B"."

`adjust.method` This is a parmeter of "topTable() in limma pacakge". Refer to the "method used to adjust the p-values for multiple testing. Options, in increasing conservatism, include "none", "BH", "BY" and "holm". See p.adjust for the complete list of options. A NULL value will result in the default adjustment method, which is "BH"."

`RNAseq` A bool type representing the input datatype is a RNASeq or not. Default is FALSE for microarray data.

Value

A list representing the differently expression for each subtype comparing to the Normal group.

Author(s)

Xu,Taosheng `<taosheng.x@gmail.com>`,Thuc Le `<Thuc.Le@unisa.edu.au>`

References

Smyth, Gordon K. "Limma: linear models for microarray data." Bioinformatics and computational biology solutions using R and Bioconductor. Springer New York, 2005. 397-420.

Examples

```
data(GeneExp)
data(miRNAExp)
GBM=list(GeneExp=GeneExp,miRNAExp=miRNAExp)
result=ExecuteSNF(GBM, clusterNum=3, K=20, alpha=0.5, t=20)
group=result$group
######Fabricate a normal group by extracting some samples from the cancer dataset
######for demonstrating the examples.
Normal_Data=GeneExp[,sample(1:100,20)]
result=DiffExp.limma(Tumor_Data=GeneExp,Normal_Data=Normal_Data,group=group,topk=NULL,RNAseq=FALSE)
```

---

drawHeatmap *Generate heatmaps*

Description

Generate heatmap for datasets.

Usage

```
drawHeatmap(
  data,
  group = NULL,
  silhouette = NULL,
  scale = "no",
  labRow = NULL,
  labCol = NULL,
  color = colorRampPalette(c("green", "black", "red"))(300),
  Title = NA
)
```

Arguments

`data` A matrix representing the genomic data such as gene expression data, miRNA expression data.
For the matrix, the rows represent the genomic features, and the columns represent the samples.

`group` A vector representing the subtype on each sample. The default is NULL. If it is not NULL, the samples will be rearrangement according to the subtypes in the heatmap.

`silhouette` An object of class silhouette. It is a result from function silhouette() or silhouette_SimilarityMatrix(). The default is NULL. If it is not NULL, an annotation will be drawn to show the silhouette width for each sample.

`scale` A string for data normalization type before heatmap drawing. The optional values are shown below:

- "no". No normalization. This is default.
- "z_score". Normalize data by z_score of features.
- "max_min". Normalize each feature by (value-min)/(max-min).

`labRow` labels for the rows. Possible values are:

	• NULL. The default value. It will use the row names of the matrix for the heatmap labels. • NA. No row label will be shown. • A list of labels.
`labCol`	labels for the columns. See labRow.
`color`	color specification for the heatmap.
`Title`	A string for the Main title of the heatmap.

Details

We applied the R package "NMF" function "aheatmap()" as the heatmap drawer.

Value

A heatmap

Author(s)

Xu,Taosheng `<taosheng.x@gmail.com>`,Thuc Le `<Thuc.Le@unisa.edu.au>`

References

Gaujoux, Renaud, and Cathal Seoighe. "A flexible R package for nonnegative matrix factorization." BMC bioinformatics 11.1 (2010): 1.

Examples

```
### SNF result analysis
data(GeneExp)
data(miRNAExp)
data(time)
data(status)
GBM=list(GeneExp=GeneExp,miRNAExp=miRNAExp)
result=ExecuteSNF(GBM, clusterNum=3, K=20, alpha=0.5, t=20)
group=result$group
distanceMatrix=result$distanceMatrix
silhouette=silhouette_SimilarityMatrix(group, distanceMatrix)
drawHeatmap(GeneExp,group,silhouette=silhouette,scale="max_min",Title="GBM Gene Expression")
drawHeatmap(GeneExp,group,silhouette=silhouette,scale="max_min",
            color="-RdYlBu",Title="GBM Gene Expression")
```

`ExecuteCC` *Execute Consensus Clustering*

Description

This function is based on the R package "ConsensusClusterPlus". We write a shell to unify the input and output format. It is helpful for the standardized flow of cancer subtypes analysis and validation. The parameters are compatible to the original R package "ConsensusClusterPlus" function "ConsensusClusterPlus()".
Please note: we add a new parameter "clusterNum" which represents the result with cancer subtypes group we want to return.

Usage

```
ExecuteCC(
  clusterNum,
  d,
  maxK = 10,
  clusterAlg = "hc",
  distance = "pearson",
  title = "ConsensusClusterResult",
  reps = 500,
  pItem = 0.8,
  pFeature = 1,
  plot = "png",
  innerLinkage = "average",
  finalLinkage = "average",
  writeTable = FALSE,
  weightsItem = NULL,
  weightsFeature = NULL,
  verbose = FALSE,
  corUse = "everything"
)
```

Arguments

`clusterNum` A integer representing the return cluster number, this value should be less than maxClusterNum(maxK). This is the only additional parameter in our function compared to the original R package "ConsensusClusterPlus". All the other parameters are compatible to the function "ConsensusClusterPlus().

`d` data to be clustered; either a data matrix where columns=items/samples and rows are features. For example, a gene expression matrix of genes in rows and microarrays in columns, or ExpressionSet object, or a distance object (only for cases of no feature resampling)

Please Note: We add a new data type (list) for this parameter. Please see details and examples.

`maxK` integer value. maximum cluster number for Consensus Clustering Algorithm to evaluate.

`clusterAlg` character value. cluster algorithm. 'hc' heirarchical (hclust), 'pam' for paritioning around medoids, 'km' for k-means upon data matrix, 'kmdist' for k-means upon distance matrices (former km option), or a function that returns a clustering.

`distance` character value. 'pearson': (1 - Pearson correlation), 'spearman' (1 - Spearman correlation), 'euclidean', 'binary', 'maximum', 'canberra', 'minkowski" or custom distance function.

`title` character value for output directory. This title can be an absolute or relative path

`reps` integer value. number of subsamples(in other words, The iteration number of each cluster number)

`pItem` Please refer to the "ConsensusClusterPlus" package for detailed information.

`pFeature` Please refer to the "ConsensusClusterPlus" package for detailed information.

`plot` Please refer to the "ConsensusClusterPlus" package for detailed information.

`innerLinkage` Please refer to the "ConsensusClusterPlus" package for detailed information.

`finalLinkage` Please refer to the "ConsensusClusterPlus" package for detailed information.

`writeTable` Please refer to the "ConsensusClusterPlus" package for detailed information.

`weightsItem` Please refer to the "ConsensusClusterPlus" package for detailed information.

`weightsFeature` Please refer to the "ConsensusClusterPlus" package for detailed information.

`verbose` Please refer to the "ConsensusClusterPlus" package for detailed information.

`corUse` Please refer to the "ConsensusClusterPlus" package for detailed information.

Details

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", we use "z-score" to normalize features for each data matrix first. Then all the normalized data matrices from the data list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.

  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **originalResult** : The clustering result of the original function "ConsensusClusterPlus()"

  Different clustering algorithms have different output formats. Although we have the group component which has consistent format for all of the algorithms (making it easy for downstream analyses), we still keep the output from the original algorithms.
- **timing** : The running time.

References

Monti, S., Tamayo, P., Mesirov, J., Golub, T. (2003) Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. Machine Learning, 52, 91-118.

See Also

`ConsensusClusterPlus`

Examples

```
### The input dataset is a single gene expression matrix.
data(GeneExp)
data(miRNAExp)
result1=ExecuteCC(clusterNum=3,d=GeneExp,maxK=10,clusterAlg="hc",distance="pearson",title="GBM")
result1$group

### The input dataset is multi-genomics data as a list
GBM=list(GeneExp=GeneExp,miRNAExp=miRNAExp)
result2=ExecuteCC(clusterNum=3,d=GBM,maxK=5,clusterAlg="hc",distance="pearson",title="GBM")
result2$group
```

`ExecuteCIMLR` *Execute CIMLR (Cancer Integration via Multikernel Learning)*

Description

CIMLR calculates the similarity between patients in multi-omic data by combining a set of Gaussian kernels for each single-omic data.

Usage

```
ExecuteCIMLR(datasets, clusterNum, k = 10, ncore = 0, plot = TRUE)
```

Arguments

`datasets` A data matrix or a list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.

`clusterNum` Number of subtypes for the samples

`k` tuning parameter in CIMLR. A default of 10 is performed.

`ncore` ratio of the number of cores to be used when computing the multi-kernel

`plot` Logical value. If true, draw the heatmap for the distance matrix with samples ordered to form clusters.

Details

The R package "CIMLR" should be installed.

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", The data matrices in the list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.

  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **timing** : The running time.

References

Daniele Ramazzotti, Avantika Lal, Bo Wang, Serafim Batzoglou, and Arend Sidow. Multi-omic tumor data reveal diversity of molecular mechanisms that correlate with survival. Nature communications, 9(1):4453, 2018.

See Also

CIMLR

Examples

```
data(GeneExp)
result=ExecuteCIMLR(GeneExp,clusterNum=5)
result$group
```

ExecuteCNMF *Execute Consensus NMF (Nonnegative matrix factorization)*

Description

Brunet applied nonnegative matrix factorization (NMF) to analyze the Gene MicroArray dataset in 2004. In the original paper, the author proved that NMF is an efficient method for distinct molecular patterns identification and provides a powerful method for class discovery. This method was implemented in an R package "NMF". Here we applied the "NMF" package to conduct the cancer subtypes identification. We write a shell to unify the input and output format. It is helpful for the standardized flow of cancer subtypes analysis and validation. The R package "NMF" should be installed.

Usage

```
ExecuteCNMF(datasets, clusterNum, nrun = 30)
```

Arguments

`datasets` A data matrix or a list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples. If the matrices have negative values, first the negative values will be set to zero to get a matrix 1; all the positive values will be set to zero to get the matrix 2; then a new matrix with all positive values will be get by concatenating matrix1 and -maxtrix2.

`clusterNum` Number of subtypes for the samples

`nrun` Number of runs to perform NMF. A default of 30 runs are performed, allowing the computation of a consensus matrix that is used in selecting the best result for cancer subtypes identification as Consensus Clustering method.

Details

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", The data matrices in the list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.

  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **originalResult** : A NMFfitX class from the result of function "nmf()".

  Different clustering algorithms have different output formats. Although we have the group component which has consistent format for all of the algorithms (making it easy for downstream analyses), we still keep the output from the original algorithms.
- **timing** : The running time.

References

[1] Brunet, Jean-Philippe, Pablo Tamayo, Todd R Golub, and Jill P Mesirov. "Metagenes and Molecular Pattern Discovery Using Matrix Factorization." Proceedings of the National Academy of Sciences 101, no. 12 (2004):4164-69.

[2] Gaujoux, Renaud, and Cathal Seoighe. "A Flexible R Package for Nonnegative Matrix Factorization." BMC Bioinformatics 11 (2010): 367. doi:10.1186/1471-2105-11-367.

See Also

`nmf`

Examples

```
data(GeneExp)
#To save the  execution time, the nrun is set to 5, but the recommended value is 30.
result=ExecuteCNMF(GeneExp,clusterNum=3,nrun=5)
result$group
```

ExecuteiCluster *Execute iCluster (Integrative clustering of multiple genomic data)*

Description

Shen (2009) proposed a latent variable regression with a lasso constraint for joint modeling of multiple omics data types to identify common latent variables that can be used to cluster patient samples into biologically and clinically relevant disease subtypes.
This function is based on the R package "iCluster". The R package "iCluster" should be installed. We write a shell to unify the input and output format. It is helpful for the standardized flow of cancer subtypes analysis and validation. The parameters is compatible to the original R package "iCluster" function "iCluster2()".
Please note: The data matrices are transposed in our function comparing to the original R package "iCluster" on the behalf of the unified input format with other functions. We try to build a standardized flow for cancer subtypes analysis and validation.

Usage

```
ExecuteiCluster(
  datasets,
  k,
  lambda = NULL,
  scale = TRUE,
  scalar = FALSE,
  max.iter = 10
)
```

Arguments

datasets	A list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples. In order to unify the input parameter with other clustering methods, the data matrices are transposed comparing to the definition in the original "iCluster" package.
k	Number of subtypes for the samples
lambda	Penalty term for the coefficient matrix of the iCluster model
scale	Logical value. If true, the genomic features in the matrix is centered.
scalar	Logical value. If true, a degenerate version assuming scalar covariance matrix is used.
max.iter	maximum iteration for the EM algorithm

Details

For iCluster algorithm, it cannot process high-dimensional data, otherwise it is very very time-consuming or reports a mistake. Based on test, it could smoothly run for the matrix with around 2000 features. Normally it need feature selection step first to reduce feature number.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).

- **originalResult** : The clustering result of the original function "iCluster2()".

  Different clustering algorithms have different output formats. Although we have the group component which has consistent format for all of the algorithms (making it easy for downstream analyses), we still keep the output from the original algorithms.

- **timing** : The running time.

References

Ronglai Shen, Adam Olshen, Marc Ladanyi. (2009). Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis. Bioinformatics 25, 2906-2912.
Ronglai Shen, Qianxing Mo, Nikolaus Schultz, Venkatraman E. Seshan, Adam B. Olshen, Jason Huse, Marc Ladanyi, Chris Sander. (2012). Integrative Subtype Discovery in Glioblastoma Using iCluster. PLoS ONE 7, e35236

See Also

`iCluster2`

Examples

```
data(GeneExp)
data(miRNAExp)
data1=FSbyVar(GeneExp, cut.type="topk",value=500)
data2=FSbyVar(miRNAExp, cut.type="topk",value=100)
GBM=list(GeneExp=data1,miRNAExp=data2)
result=ExecuteiCluster(datasets=GBM, k=3, lambda=list(0.44,0.33,0.28))
result$group
```

`ExecuteIntClust` *Execute IntClust IntClust is a integrative method to classify samples to ten breast cancer subtypes.*

Description

IntClust applies iCluster on a matched mRNA-CNV breast cancer dataset with 997 samples and identifies ten breast cancer subtypes (so-called integrative subtypes). Similar to PAM50, IntClust builds three centroid-based predictors based on 612 cis-eQTLs gene drivers by using PAM. The R package "genefu" should be installed.

Usage

```
ExecuteIntClust(datasets)
```

Arguments

`datasets`	A data matrix or a list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.

Details

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", The data matrices in the list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **timing** : The running time.

References

[1] Christina Curtis, Sohrab P Shah, Suet-Feung Chin, Gulisa Turashvili, Oscar M Rueda, Mark J Dunning, Doug Speed, Andy G Lynch, Shamith Samarajiwa, Yinyin Yuan, et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. Nature, 486(7403):346, 2012.

[2] H Raza Ali, Oscar M Rueda, Suet-Feung Chin, Christina Curtis, Mark J Dunning, Samuel AJR Aparicio, and Carlos Caldas. Genome-driven integrated classification of breast cancer validated in over 7,500 samples. Genome biology, 15(8):431, 2014.

See Also

molecular.subtyping

Examples

```
data(GeneExp)
result=ExecuteIntClust(GeneExp)
result$group
```

ExecuteintNMF *Execute intNMF (an integrative approach for cancer subtype classification based on non-negative matrix factorization)*

Description

intNMF is an integrative approach for cancer subtype classification based on non-negative matrix factorization.

Usage

```
ExecuteintNMF(
  datasets,
  clusterNum,
  maxiter = 150,
  st.count = 20,
  n.ini = 15,
  plot = TRUE
)
```

Arguments

datasets	A data matrix or a list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.
clusterNum	Number of subtypes for the samples
maxiter	Maximum number of iteration, default is 150.
st.count	Count for stability in connectivity matrix, default is 20.
n.ini	Number of initializations of the random matrices, default is 15.
plot	Logical value. If true, draw the heatmap for the distance matrix with samples ordered to form clusters.

Details

The R package "IntNMF" should be installed.

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", The data matrices in the list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.

  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **timing** : The running time.

References

Chalise, Prabhakar, and Brooke L. Fridley. "Integrative clustering of multi-level ‘omic data based on non-negative matrix factorization algorithm." PloS one 12.5 (2017): e0176278.

Examples

```
data(GeneExp)
result=ExecuteintNMF(list(GeneExp),clusterNum=3)
result$group
```

`ExecuteNEMO` *Execute NEMO (NEighborhood based Multi-Omics clustering)*

Description

NEMO is a similarity-based multi-omic clustering method based on the radial basis of function kernel and spectral clustering method. Different from other multi-omic clustering methods, NEMO can apply to partial data that some omics may not be measured for some patients. NEMO is simple, and faster than other multi-omics clustering methods but achieved the comparable performance to others.

Usage

```
ExecuteNEMO(datasets, clusterNum, num.neighbors = 50, plot = TRUE)
```

Arguments

`datasets` A data matrix or a list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.

`clusterNum` Number of subtypes for the samples `num.neighbors` The number of neighbors to use for each omic. It can either be a number, a list of numbers or NA. If it is a number, this is the number of neighbors used for all omics. If this is a list, the number of neighbors are taken for each omic from that list. If it is NA, each omic chooses the number of neighbors to be the number of samples divided by NUM.NEIGHBORS.RATIO.

`plot` Logical value. If true, draw the heatmap for the distance matrix with samples ordered to form clusters.

Details

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", The data matrices in the list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.

  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **timing** : The running time.

References

Nimrod Rappoport and Ron Shamir. Nemo: cancer subtyping by integration of partial multi-omic data. Bioinformatics, 35(18):3348-3356, 2019.

Examples

```
data(GeneExp)
result=ExecuteNEMO(GeneExp,clusterNum=3)
result$group
```

ExecutePAM50 *Execute PAM50 classifier*

Description

PAM50 is a gene-based method to classify samples to the five subtypes: Basal, Luminal A, Luminal B, Her2-enriched and Normal-like. PAM50 constructs a centroid-based predictor by using the Prediction Analysis of Microarray (PAM) algorithm on 50 gene signatures. The R package "genefu" should be installed.

Usage

```
ExecutePAM50(datasets)
```

Arguments

datasets	A data matrix or a list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.

Details

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", The data matrices in the list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **timing** : The running time.

References

Joel S Parker, Michael Mullins, Maggie CU Cheang, Samuel Leung, David Voduc, Tammi Vickery, Sherri Davies, Christiane Fauron, Xiaping He, Zhiyuan Hu, et al. Supervised risk predictor of breast cancer based on intrinsic subtypes. Journal of clinical oncology, 27(8):1160, 2009.

See Also

molecular.subtyping

Examples

```
data(GeneExp)
result=ExecutePAM50(GeneExp)
result$group
```

`ExecutePINS` *Execute PINS (Perturbation clustering for data INtegration and disease Subtyping)*

Description

PINS initially clusters patients based on each omic data separately using perturbation clustering and outputs the optimal number of clusters ki, original connectivity matrix Ci and the perturbed connectivity matrix Ai, where i is the index of i-th omic data. The R package "PINSPlus" should be installed.

Usage

```
ExecutePINS(
  datasets,
  clusterNum,
  ncore = 1,
  clusteringMethod = "kmeans",
  perturbMethod = "noise",
  iterMin = 20,
  iterMax = 200,
  madMin = 0.001,
  msdMin = 1e-06
)
```

Arguments

`datasets` A data matrix or a list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.

`clusterNum` Number of subtypes for the samples

`ncore` Number of cores that the algorithm should use. Default value is 1.

`clusteringMethod` The name of built-in clustering algorithm that PerturbationClustering will use. Currently supported algorithm are kmeans, pam and hclust. Default value is "kmeans".

`perturbMethod` The clustering algorithm function that will be used instead of built-in algorithms.

`iterMin` The minimum number of iterations. Default value is 20.

`iterMax` The maximum number of iterations. Default value is 200.

`madMin` The minimum of Mean Absolute Deviation of AUC of Connectivity matrix for each k. Default value is 1e-03.

`msdMin` The minimum of Mean Square Deviation of AUC of Connectivity matrix for each k. Default value is 1e-06.

Details

If the data is a list containing the matched mutli-genomics data matrices like the input as "ExecuteiCluster()" and "ExecuteSNF()", The data matrices in the list are concatenated according to samples. The concatenated data matrix is the samples with a long features (all features in the data list). Our purpose is to make convenient comparing the different method with same dataset format. See examples.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **timing** : The running time.

References

Tin Nguyen, Rebecca Tagett, Diana Diaz, and Sorin Draghici. A novel approach for data integration and disease subtyping. Genome research, 27(12):2025-2039, 2017.

See Also

PerturbationClustering

Examples

```
data(GeneExp)
result=ExecutePINS(GeneExp,clusterNum=5)
result$group
```

ExecuteSNF *Execute SNF(Similarity Network Fusion)*

Description

SNF is a multi-omics data processing method that constructs a fusion patient similarity network by integrating the patient similarity obtained from each of the genomic data types. SNF calculates the similarity between patients using each single data type separately. The similarities between patients from different data types are then integrated by a cross-network diffusion process to construct the fusion patient similarity matrix. Finally, a clustering method is applied to the fusion patient similarity matrix to cluster patients into different groups, which imply different cancer subtypes. This function is based on the R package "SNFtool". The R package "SNFtool" should be installed. We write a function to integrate the clustering process and unify the input and output format. It is helpful for the standardized flow of cancer subtypes analysis and validation.
Please note: The data matrices are transposed in our function comparing to the original R package "SNFtools". We try to build a standardized flow for cancer subtypes analysis and validation.

Usage

```
ExecuteSNF(datasets, clusterNum, K = 20, alpha = 0.5, t = 20, plot = TRUE)
```

Arguments

`datasets`	A list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.
`clusterNum`	A integer representing the return cluster number
`K`	Number of nearest neighbors
`alpha`	Variance for local model
`t`	Number of iterations for the diffusion process
`plot`	Logical value. If true, draw the heatmap for the distance matrix with samples ordered to form clusters.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.
  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **originalResult** : The clustering result of the original SNF algorithm"
  Different clustering algorithms have different output formats. Although we have the group component which has consistent format for all of the algorithms (making it easy for downstream analyses), we still keep the output from the original algorithms.
- **timing** : The running time.

References

B Wang, A Mezlini, F Demir, M Fiume, T Zu, M Brudno, B Haibe-Kains, A Goldenberg (2014) Similarity Network Fusion: a fast and effective method to aggregate multiple data types on a genome wide scale. Nature Methods. Online. Jan 26, 2014

See Also

`affinityMatrix` `SNF`

Examples

```
data(GeneExp)
data(miRNAExp)
GBM=list(GeneExp=GeneExp,miRNAExp=miRNAExp)
result=ExecuteSNF(GBM, clusterNum=3, K=20, alpha=0.5, t=20)
result$group
```

ExecuteSNF.CC *Execute the combined SNF (Similarity Network Fusion) and Consensus clustering*

Description

This function is a combined process of SNF and Consensus Clustering for cancer subtypes identification. First it applied SNF to get the fusion patients similarity matrix. Then use this fusion patients similarity matrix as the sample distance for Consensus Clustering.

Usage

```
ExecuteSNF.CC(
  datasets,
  clusterNum,
  K = 20,
  alpha = 0.5,
  t = 20,
  maxK = 10,
  pItem = 0.8,
  reps = 500,
  title = "ConsensusClusterResult",
  plot = "png",
  finalLinkage = "average"
)
```

Arguments

`datasets`	A list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples. Same as ExecuteSNF
`clusterNum`	A integer representing the return cluster number. Same as ExecuteSNF
`K`	Number of nearest neighbors. Same as ExecuteSNF
`alpha`	Variance for local model. Same as ExecuteSNF
`t`	Number of iterations for the diffusion process. Same as ExecuteSNF
`maxK`	integer value. maximum cluster number for Consensus Clustering Algorithm to evaluate. Same as ExecuteCC.
`pItem`	Same as ExecuteCC
`reps`	integer value. number of subsamples(in other words, The iteration number of each cluster number). Same as ExecuteCC
`title`	character value for output directory. This title can be an absolute or relative path. Same as ExecuteCC
`plot`	Same as ExecuteCC
`finalLinkage`	Same as ExecuteCC

Value

Same as the ExecuteCC(). A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).
- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.

  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **originalResult** : The clustering result of the original function "ConsensusClusterPlus()"

  Different clustering algorithms have different output formats. Although we have the group component which has consistent format for all of the algorithms (making it easy for downstream analyses), we still keep the output from the original algorithms.
- **timing** : The running time.

See Also

`ExecuteSNF` `ExecuteCC`

Examples

```
data(GeneExp)
data(miRNAExp)
GBM=list(GeneExp,miRNAExp)
result=ExecuteSNF.CC(GBM, clusterNum=3, K=20, alpha=0.5, t=20,
                    maxK = 5, pItem = 0.8,reps=500,
                    title = "GBM", plot = "png",
                    finalLinkage ="average")
result$group
```

`ExecuteWSNF` *Execute the WSNF(Weighted Similarity Network Fusion)*

Description

WSNF is a caner subtype identificaton method with the assistance of the gene regulatory network information. The basic idea of the WSNF is to set the different regulatory importance(ranking) for each feature. In the WSNF manuscript, WSNF makes use of the miRNA-TF-mRNA regulatory network to take the importance of the features into consideration.

Usage

```
ExecuteWSNF(
  datasets,
  feature_ranking,
  beta = 0.8,
  clusterNum,
  K = 20,
  alpha = 0.5,
  t = 20,
  plot = TRUE
)
```

Arguments

`datasets` A list containing data matrices. For each data matrix, the rows represent genomic features, and the columns represent samples.

`feature_ranking`
A list containing numeric vetors. The length of the feature_ranking list should equal to the length of datasets list. For each numeric vetor represents the ranking of each feature in the corresponding data matrix. The order of the ranking should also mathch the order of the features in the corresponding data matrix. We proive a ranking list for most mRNA, TF(transcription factor) and miRNA features. The ranking for features caculated based on the miRNA-TF-miRNA regulatory network which was promoted in our published work: Identifying Cancer Subtypes from miRNA-TF-mRNA Regulatory Networks and Expression Data(PLos One,2016).

`beta` A tuning parameter for the feature_ranking contributes the weight of each feature.
A linear model is applied to integrate feature_ranking and MAD(median absolute deviation) to generated the final weight for each feature using for the algorithm. The final weight is cauculated as the formula below:
Weight(f_i)=beta * feature_ranking + (1-beta) MAD(f_i)

`clusterNum` A integer representing the return cluster number

`K` Number of nearest neighbors

`alpha` Variance for local model

`t` Number of iterations for the diffusion process

`plot` Logical value. If true, draw the heatmap for the distance matrix with samples ordered to form clusters.

Value

A list with the following elements.

- **group** : A vector represent the group of cancer subtypes. The order is corresponding to the the samples in the data matrix.

  This is the most important result for all clustering methods, so we place it as the first component. The format of group is consistent across different algorithms and therefore makes it convenient for downstream analyses. Moreover, the format of group is also compatible with the K-means result and the hclust (after using the cutree() function).

- **distanceMatrix** : It is a sample similarity matrix. The more large value between samples in the matrix, the more similarity the samples are.

  We extracted this matrix from the algorithmic procedure because it is useful for similarity analysis among the samples based on the clustering results.
- **originalResult** : The clustering result of the original SNF algorithm.

  Different clustering algorithms have different output formats. Although we have the group component which has consistent format for all of the algorithms (making it easy for downstream analyses), we still keep the output from the original algorithms.
- **timing** : The running time.

References

Xu, T., Le, T. D., Liu, L., Wang, R., Sun, B., & Li, J. (2016). Identifying cancer subtypes from mirna-tf-mrna regulatory networks and expression data. PloS one, 11(4), e0152792.

See Also

ExecuteSNF

Examples

```
data(GeneExp)
data(miRNAExp)
GBM=list(GeneExp,miRNAExp)
###1. Use the defualt ranking in the package.
data(Ranking)
####Retrieve the feature ranking for genes
gene_Name=rownames(GeneExp)
index1=match(gene_Name,Ranking$mRNA_TF_miRNA.v21_SYMBOL)
gene_ranking=data.frame(gene_Name,Ranking[index1,],stringsAsFactors=FALSE)
index2=which(is.na(gene_ranking$ranking_default))
gene_ranking$ranking_default[index2]=min(gene_ranking$ranking_default,na.rm =TRUE)

####Retrieve the feature ranking for miRNAs
miRNA_ID=rownames(miRNAExp)
index3=match(miRNA_ID,Ranking$mRNA_TF_miRNA_ID)
miRNA_ranking=data.frame(miRNA_ID,Ranking[index3,],stringsAsFactors=FALSE)
index4=which(is.na(miRNA_ranking$ranking_default))
miRNA_ranking$ranking_default[index4]=min(miRNA_ranking$ranking_default,na.rm =TRUE)
###Clustering
ranking1=list(gene_ranking$ranking_default ,miRNA_ranking$ranking_default)
result1=ExecuteWSNF(datasets=GBM, feature_ranking=ranking1, beta = 0.8, clusterNum=3,
                   K = 20,alpha = 0.5, t = 20, plot = TRUE)

###2. User input ranking
# Fabricate a ranking list for demonstrating the examples.
ranking2=list(runif(nrow(GeneExp), min=0, max=1),runif(nrow(miRNAExp), min=0, max=1))
result2=ExecuteWSNF(datasets=GBM, feature_ranking=ranking2, beta = 0.8, clusterNum=3,
                   K = 20,alpha = 0.5, t = 20, plot = TRUE)
```

FSbyCox — *Biological feature (such as gene) selection based on Cox regression model.*

Description

Cox model (Proportional hazard model) is a statistical approach for survival risk analysis. We applied the univariate Cox model for feature selection. The proportional hazard assumption test is used to evaluate the significant level of each biological feature related to the survival result for samples. Eventually, the most significant genes are selected for clustering analysis.

Usage

```
FSbyCox(Data, time, status, cutoff = 0.05)
```

Arguments

`Data` A data matrix representing the genomic data measured in a set of samples. For the matrix, the rows represent the genomic features, and the columns represent the samples.

`time` A numeric vector representing the survival time (days) of a set of samples. Note that the order of the time should map the samples in the Data matrix.

`status` A numeric vector representing the survival status of a set of samples. 0=alive/censored, 1=dead. Note that the order of the time should map the samples in the Data matrix.

`cutoff` A numeric value in (0,1) representing whether the significant feature Xi is selected according to the Proportional Hazards Assumption p-value of the feature Xi. If p-value(Xi)<cutoff, the features Xi will be selected for downstream analysis. Normally the significant level is set to 0.05.

Value

A data matrix, extracted a subset with significant features from the input data matrix. The rows represent the significant features, and the columns represents the samples.

Author(s)

Xu,Taosheng <`taosheng.x@gmail.com`>, Thuc Le <`Thuc.Le@unisa.edu.au`>

References

Andersen, P. and Gill, R. (1982). Cox's regression model for counting processes, a large sample study. Annals of Statistics 10, 1100-1120.
Therneau, T., Grambsch, P., Modeling Survival Data: Extending the Cox Model. Springer-Verlag, 2000.

Examples

```
data(GeneExp)
data(time)
data(status)
data1=FSbyCox(GeneExp,time,status,cutoff=0.05)
```

FSbyMAD *Biological feature (such as gene) selection based on the most variant Median Absolute Deviation (MAD).*

Description

Biological feature (such as gene) selection based on the most variant Median Absolute Deviation (MAD).

Usage

```
FSbyMAD(Data, cut.type = "topk", value)
```

Arguments

`Data`	A data matrix representing the genomic data measured in a set of samples. For the matrix, the rows represent the genomic features, and the columns represents the samples.
`cut.type`	A character value representing the selection type. The optional values are shown below: • "topk" • "cutoff"
`value`	A numeric value. If the cut.type="topk", the top number of value features are selected. If the cut.type="cutoff", the features with (MAD>value) are selected.

Value

An extracted subset data matrix with the most variant MAD features from the input data matrix.

Author(s)

Xu,Taosheng <`taosheng.x@gmail.com`>, Thuc Le <`Thuc.Le@unisa.edu.au`>

Examples

```
data(GeneExp)
data1=FSbyMAD(GeneExp, cut.type="topk",value=1000)
```

FSbyPCA — *Biological feature (such as gene) dimension reduction and extraction based on Principal Component Analysis.*

Description

This function is based on the prcomp(), we write a shell for it and make it easy to use on genomic data.

Usage

```
FSbyPCA(Data, PC_percent = 1, scale = TRUE)
```

Arguments

`Data`	A data matrix representing the genomic data measured in a set of samples. For the matrix, the rows represent the genomic features, and the columns represents the samples.
`PC_percent`	A numeric values in [0,1] representing the ratio of principal component is seclected.
`scale`	A bool variable, If true, the Data is normalized before PCA.

Value

A new matrix with full or part Principal Component in new projection space.

Author(s)

Xu,Taosheng <taosheng.x@gmail.com>,Thuc Le <Thuc.Le@unisa.edu.au>

Examples

```
data(GeneExp)
data1=FSbyPCA(GeneExp, PC_percent=0.9,scale = TRUE)
```

FSbyVar — *Biological feature (such as gene) selection based on the most variance.*

Description

Biological feature (such as gene) selection based on the most variance.

Usage

```
FSbyVar(Data, cut.type = "topk", value)
```

Arguments

`Data` A data matrix representing the genomic data measured in a set of samples. For the matrix, the rows represent the genomic features, and the columns represents the samples.

`cut.type` A character value representing the selection type. The optional values are shown below:

- "topk"
- "cutoff"

`value` A numeric value.
If the cut.type="topk", the top number of value features are selected.
If the cut.type="cutoff", the features with (var>value) are selected.

Value

An extracted subset data matrix with most variance features from the input data matrix.

Author(s)

Xu,Taosheng <taosheng.x@gmail.com>, Thuc Le <Thuc.Le@unisa.edu.au>

Examples

```
data(GeneExp)
data1=FSbyVar(GeneExp, cut.type="topk",value=1000)
```

GeneExp *Dataset: Gene expression*

Description

A glioblastoma (GBM) gene expression dataset downloaded from TCGA. This is a small dataset with 1500 genes and 100 cancer samples extracted from gene expression data for examples.

Format

A data matrix

Details

- Rows are genes
- Columns are cancer samples

Examples

```
data(GeneExp)
```

getFilePath *get file path based on a given folder name and a given file name*

Description

get file path based on a given folder name and a given file name

Usage

```
getFilePath(foldername = NULL, filename = NULL)
```

Arguments

`foldername`	A given folder name; if NULL, will set to the current working directory.
`filename`	A given file name

Value

The full path of the file

getMeanSilhouette *get mean Silhouette from predicted subtypes*

Description

get mean Silhouette from predicted subtypes

Usage

```
getMeanSilhouette(group, distanceMatrix)
```

Arguments

`group`	predicted subtypes of samples
`distanceMatrix`	A distace matrix of samples

Value

The mean Silhouette width

Examples

```
## Not run:
res = getMeanSilhouette(IC10_res$group, distanceMatrix)

## End(Not run)
```

getPvalue *get p-value of the Log-rank test from predicted subtypes*

Description

get p-value of the Log-rank test from predicted subtypes

Usage

```
getPvalue(time, status, group)
```

Arguments

`time`	survival time of samples
`status`	event status of samples
`group`	predicted subtypes of samples

Value

The p-value of the Log-rank test

Examples

```
## Not run:
res = getPvalue(dclin[,1],dclin[,2],IC10_res$group)

## End(Not run)
```

lncRNA12 *Dataset: lncRNA signatures*

Description

12 lncRNA signatures

Format

A data frame

Details

- Rows are lncRNAs

Examples

```
data(sig.lncRNA12)
```

lncRNA12model *Evaluate Cancer Prognosis based on 12 lncRNA signatures Cancer Prognosis to compute the risk scores for lncRNA data based on 12 lncRNA signatures*

Description

Evaluate Cancer Prognosis based on 12 lncRNA signatures Cancer Prognosis to compute the risk scores for lncRNA data based on 12 lncRNA signatures

Usage

```
lncRNA12model(data, annot, RNASignature = NULL, do.mapping = FALSE)
```

Arguments

`data`	data to be computed for cancer Prognosis risk scores; a data matrix, rows= lncRNA annotated with lncRNA names and columns=terms/samples.
`annot`	annot holds RNA names in the data.
`RNASignature`	RNA signatures, if none, will import the 12 lncRNA signatures.
`do.mapping`	if do.mapping equals Ture, use probe IDs instead of gene symbols.

Details

The 12 lncRNA signatures are "RP1-34M23.5","RP11-202K23.1", "RP11-560G2.1", "RP4-591L5.2","RP13-104F24.2", "RP11-506D12.5","ERVH48-1","RP4-613B23.1", "RP11-360F5.1" ,"CTD-2031P19.5", "RP11-247A12.8", "SNHG7".

Value

A Numeric Vector

References

- **lncRNA12model**: Zhou M, Zhong L, Xu W, Sun Y, Zhang Z, Zhao H, Yang L, Sun J. Discovery of potential prognostic long non-coding RNA biomarkers for predicting the risk of tumor recurrence of breast cancer patients. Sci Rep. 2016;6:31038.

Examples

```
data(TCGA500)
data = exprs(TCGA500)
annot = fData(TCGA500)
res =  lncRNA12model(data,annot)
```

lncRNA5 *Dataset: lncRNA signatures*

Description

5 lncRNA signatures

Format

A data frame

Details

- Rows are lncRNAs

Examples

```
data(sig.lncRNA5)
```

lncRNA5model *Evaluate Cancer Prognosis based on 5 lncRNA signatures Cancer Prognosis to compute the risk scores for lncRNA data based on 5 lncRNA signatures*

Description

Evaluate Cancer Prognosis based on 5 lncRNA signatures Cancer Prognosis to compute the risk scores for lncRNA data based on 5 lncRNA signatures

Usage

```
lncRNA5model(data, annot, RNASignature = NULL, do.mapping = FALSE)
```

Arguments

`data`	data to be computed for cancer Prognosis risk scores; a data matrix, rows= lncRNA annotated with lncRNA names and columns=terms/samples.
`annot`	annot holds RNA names in the data.
`RNASignature`	RNA signatures, if none, will import the 5 lncRNA signatures.
`do.mapping`	if do.mapping equals Ture, use probe IDs instead of gene symbols.

Details

The 5 lncRNA signatures are "RP11-524D16-A.3", "HOTAIR", "AL645608.1", "TSPOAP1-AS1", "RP11-13L2.4".

Value

A Numeric Vector

References

- **lncRNA5model**: Li J, Wang W, Xia P, et al. Identification of a five-lncRNA signature for predicting the risk of tumor recurrence in patients with breast cancer. Int J Cancer. 2018; 143: 2150-2160.

Examples

```
data(TCGA500)
data = exprs(TCGA500)
annot = fData(TCGA500)
res =  lncRNA5model(data,annot)
```

`lncRNA6` *Dataset: lncRNA signatures*

Description

6 lncRNA signatures

Format

A data frame

Details

- Rows are lncRNAs

Examples

```
data(sig.lncRNA6)
```

`lncRNA6model` *Evaluate Cancer Prognosis based on 6 lncRNA signatures Cancer Prognosis to compute the risk scores for lncRNA data based on 6 lncRNA signatures*

Description

Evaluate Cancer Prognosis based on 6 lncRNA signatures Cancer Prognosis to compute the risk scores for lncRNA data based on 6 lncRNA signatures

Usage

```
lncRNA6model(data, annot, RNASignature = NULL, do.mapping = FALSE)
```

Arguments

`data`	data to be computed for cancer Prognosis risk scores; a data matrix, rows= lncRNA annotated with lncRNA names and columns=terms/samples.
`annot`	annot holds RNA names in the data.
`RNASignature`	RNA signatures, if none, will import the 6 lncRNA signatures.
`do.mapping`	if do.mapping equals Ture, use probe IDs instead of gene symbols.

Details

The 6 lncRNA signatures are "HAGLR","STK4-AS1", "DLEU7-AS1", "LINC00957","LINC01614", "ITPR1-AS1".

Value

A Numeric Vector

References

- **lncRNA6model**: Zhong L, Lou G, Zhou X, Qin Y, Liu L, Jiang W. A six-long non-coding RNAs signature as a potential prognostic marker for survival prediction of ER-positive breast cancer patients. Oncotarget. 2017;8(40):67861.

Examples

```
data(TCGA500)
data = exprs(TCGA500)
annot = fData(TCGA500)
res =  lncRNA6model(data,annot)
```

`loadData` *load R data from a given directory*

Description

load R data from a given directory

Usage

```
loadData(filefolder = NULL, dataSets = NULL)
```

Arguments

`filefolder`	A number
`dataSets`	names of datasets; a string vector.

Examples

```
## Not run:
dn = c("TCGA","UK","HEL","GSE19783")
loadData("data",dn)

## End(Not run)
```

LogRank *Evaluate the predicted risk scores from benchmark methods*

Description

Evaluate the predicted risk scores from benchmark methods

Usage

```
LogRank(data, survival)
```

Arguments

`data`	A numeric data frame or metrix indicating the predicted risk scores from benchmark methods
`survival`	A data frame holds survival time and event status

Value

The p-values of the benchmark methods

See Also

`survdiff`

Examples

```
## Not run:
res = LogRank(data=resMatrix[[i]], survival)

## End(Not run)
```

miRNA10 *Dataset: miRNA signatures*

Description

The 10 miRNAs signatures are "hsa-miR-144", "hsa-miR-150", "hsa-miR-210", "hsa-miR-27b", "hsa-miR-30c", "hsa-miR-342" "hsa-miR-128a", "hsa-miR-135a", "hsa-miR-767-3p", "hsa-miR-769-3p". Note that the miRbase version of miRNAs signatures is v9_2, please check the miRbase version of your own dataset.

Format

A data frame

Details

- Rows are miRNAs

Examples

```
data(sig.miRNA10)
```

miRNA10model *Evaluate Cancer Prognosis based on 10 miRNA signatures Cancer Prognosis to compute the risk scores for miRNA data based on 10 miRNA signatures*

Description

Evaluate Cancer Prognosis based on 10 miRNA signatures Cancer Prognosis to compute the risk scores for miRNA data based on 10 miRNA signatures

Usage

```
miRNA10model(data, annot, RNASignature = NULL)
```

Arguments

`data`	data to be computed for cancer Prognosis risk scores; a data matrix, rows= miRNAs annotated with miRNA names and columns=terms/samples.
`annot`	annot holds RNA names in the data.
`RNASignature`	RNA signatures, if none, will import the 10 microRNA signatures.

Details

The 10 miRNAs signatures are "hsa-miR-144", "hsa-miR-150", "hsa-miR-210", "hsa-miR-27b", "hsa-miR-30c", "hsa-miR-342" "hsa-miR-128a", "hsa-miR-135a", "hsa-miR-767-3p", "hsa-miR-769-3p". Note that the miRbase version of miRNAs signatures is v9_2, please check the miRbase version of your own dataset.

Value

A Numeric Vector

References

- **miRNA10model**: Buffa, F. M. et al. microRNA associated progression pathways and potential therapeutic targets identified by integrated mRNA and microRNA expression profiling in breast cancer. Cancer Res. 71, 5635 (2011).

Examples

```
data(TCGA500)
data = exprs(TCGA500)
annot = fData(TCGA500)
res =  miRNA10model(data,annot)
```

`miRNAExp` *Dataset: miRNA expression*

Description

A glioblastoma (GBM) miRNA expression dataset downloaded from TCGA. This is a small miRNA expression dataset with 470 miRNAs and 100 cancer samples extracted from miRNA expression data for examples.

Format

A data matrix

Details

- Rows are miRNAs
- Columns are cancer samples

Examples

```
data(miRNAExp)
```

`Ranking` *Dataset: A default ranking of features for the fuction ExecuteWSNF()*

Description

A dataframe represents the regulatory ranking for features(mRNA,TF,miRNA) caculated based on the miRNA-TF-miRNA regulatory network which was promoted in our published work: Identifying Cancer Subtypes from miRNA-TF-mRNA Regulatory Networks and Expression Data(PLos One,2016).

Format

dataframe

Details

- mRNA_TF_miRNA_ID : ENTREZID for genes(mRNA,TF) and miRBase Accession ID for miRNAs.
- mRNA_TF_miRNA.v21._SYMBOL: gene symbol and miRNA names(miRBase Version 21)
- feature_ranking: the numeric values represents regulatory ranking for each feature.

References

Xu, T., Le, T. D., Liu, L., Wang, R., Sun, B., & Li, J. (2016). Identifying cancer subtypes from mirna-tf-mrna regulatory networks and expression data. PloS one, 11(4), e0152792.

Examples

```
data(Ranking)
```

RNAmodel — *Evaluate Cancer Prognosis based on 37 miRNA/mRNA signatures Cancer Prognosis to compute the risk scores for miRNA/mRNA data based on 37 microRNA/mRNA signatures*

Description

Evaluate Cancer Prognosis based on 37 miRNA/mRNA signatures Cancer Prognosis to compute the risk scores for miRNA/mRNA data based on 37 microRNA/mRNA signatures

Usage

```
RNAmodel(data, annot, RNASignature = NULL)
```

Arguments

`data`	data to be computed for cancer Prognosis risk scores; a data matrix, rows= miRNAs annotated with miRNA names and columns=terms/samples.
`annot`	annot holds miRNA/mRNA names in the data.
`RNASignature`	RNA signatures, if none, will import the 37 microRNA/mRNA signatures.

Details

The 7 miRNAs signatures are "hsa-miR-103", "hsa-miR-1307", "hsa-miR-148b", "hsa-miR-328", "hsa-miR-484", "hsa-miR-874","hsa-miR-93". The 30 mRNA signatures are "ACSL1" ,"ADAT1", "ANKRD52","BIRC6", "CPT1A", "CXCR7","DAAM1", "DIP2B", "FAM199X", "FAM91A1", "FRZB", "GLA", "GMCL1", "HRASLS", "HSP90AA1", "MCM10", "ME1", "NDRG1", "NOTCH2NL","OTUD6B","P
"PGK1", "PIK3CA", "PTAR1", "SMG1", "TRIM23", "TTC3", "UBR5", "UBXN7", "ZFC3H1". Note that the miRbase version of miRNAs signatures is v16, please check the miRbase version of your own dataset.

Value

A Numeric Vector

References

- **RNAmodel**: Volinia S, Croce CM. Prognostic microRNA/mRNA signature from the integrated analysis of patients with invasive breast cancer. Proc Natl Acad Sci USA. 2013;110:7413-7417.

Examples

```
data(TCGA500)
data = exprs(TCGA500)
annot = fData(TCGA500)
res =  RNAmodel(data,annot)
```

RNASig — *Dataset: RNA signatures*

Description

The 7 miRNAs signatures are "hsa-miR-103", "hsa-miR-1307", "hsa-miR-148b", "hsa-miR-328", "hsa-miR-484", "hsa-miR-874","hsa-miR-93". The 30 mRNA signatures are "ACSL1" ,"ADAT1", "ANKRD52","BIRC6", "CPT1A", "CXCR7","DAAM1", "DIP2B", "FAM199X", "FAM91A1", "FRZB", "GLA", "GMCL1", "HRASLS", "HSP90AA1", "MCM10", "ME1", "NDRG1", "NOTCH2NL","OTUD6B","P "PGK1", "PIK3CA", "PTAR1", "SMG1", "TRIM23", "TTC3", "UBR5", "UBXN7", "ZFC3H1". Note that the miRbase version of miRNAs signatures is v16, please check the miRbase version of your own dataset.

Format

A data frame

Details

- Rows are RNAs

Examples

```
data(sig.RNA37)
```

saveFigure — *This function save the figure in the current plot.*

Description

This function save the figure in the current plot.

Usage

```
saveFigure(
  foldername = NULL,
  filename = "saveFig",
  image_width = 10,
  image_height = 10,
  image_res = 300
)
```

Arguments

foldername	Character values. It specifies the folder name which will be created in the present working path.
filename	Character values. It specifies the saved file name.
image_width	the figure width
image_height	the figure height
image_res	the figure resolution

Value

A * .png file in the specified folder.

Author(s)

Xu,Taosheng <taosheng.x@gmail.com>,Thuc Le <Thuc.Le@unisa.edu.au>

Examples

```
data(GeneExp)
data(miRNAExp)
data(time)
data(status)
GBM=list(GeneExp=GeneExp,miRNAExp=miRNAExp)
result=ExecuteSNF(GBM, clusterNum=3, K=20, alpha=0.5, t=20)
group=result$group
distanceMatrix=result$distanceMatrix
p_value=survAnalysis(mainTitle="GBM",time,status,group,
      distanceMatrix=distanceMatrix,similarity=TRUE)
saveFigure(foldername="GBM",filename="GBM",image_width=10,image_height=10,image_res=300)
```

sigclustTest *A statistical method for testing the significance of clustering results.*

Description

SigClust (Statistical significance of clustering) is a statistical method for testing the significance of clustering results. SigClust can be applied to assess the statistical significance of splitting a data set into two clusters. SigClust studies whether clusters are really there, using the 2-means (k = 2) clustering index as a statistic. It assesses the significance of clustering by simulation from a single null Gaussian distribution. Null Gaussian parameters are estimated from the data. Here we apply the SigClust to assess the statistical significance of pairwise subtypes. "sigclust" package should be installed.

Usage

```
sigclustTest(Data, group, nsim = 1000, nrep = 1, icovest = 1)
```

Arguments

Data	A data matrix representing the genomic data measured in a set of samples. For the matrix, the rows represent the genomic features, and the columns represents the samples.
group	The subtypes label of each sample
nsim	This is a parameter inherited from sigclust() in "sigclust" Package. Number of simulated Gaussian samples to estimate the distribution of the clustering index for the main p-value computation.
nrep	This is a parameter inherited from sigclust() in "sigclust" Package. Number of steps to use in 2-means clustering computations (default=1, chosen to optimize speed).

`icovest` This is a parameter inherited from sigclust() in "sigclust" Package. Covariance estimation type: 1. Use a soft threshold method as constrained MLE (default); 2. Use sample covariance estimate (recommended when diagnostics fail); 3. Use original background noise threshold estimate (from Liu, et al, (2008)) ("hard thresholding").

Value

A matrix indicates the p-value between pairwise subtypes.

Author(s)

Xu,Taosheng <`taosheng.x@gmail.com`>,Thuc Le <`Thuc.Le@unisa.edu.au`>

References

Liu, Yufeng, Hayes, David Neil, Nobel, Andrew and Marron, J. S, 2008, Statistical Significance of Clustering for High-Dimension, Low-Sample Size Data, Journal of the American Statistical Association 103(483) 1281-1293.
Huang, Hanwen, Yufeng Liu, Ming Yuan, and J. S. Marron. "Statistical Significance of Clustering Using Soft Thresholding." Journal of Computational and Graphical Statistics, no. just-accepted (2014): 00-00.

See Also

`sigclust`

Examples

```
data(GeneExp)
data(miRNAExp)
data(time)
data(status)
GBM=list(GeneExp=GeneExp,miRNAExp=miRNAExp)
result=ExecuteSNF(GBM, clusterNum=3, K=20, alpha=0.5, t=20)
group=result$group
sigclust1=sigclustTest(miRNAExp,group, nsim=500, nrep=1, icovest=3)
sigclust2=sigclustTest(miRNAExp,group, nsim=1000, nrep=1, icovest=1)
```

`silhouette_SimilarityMatrix` *Compute or Extract Silhouette Information from Clustering based on similarity matrix.*

Description

Silhouette refers to a method of interpretation and validation of consistency within clusters of data. The technique provides a succinct graphical representation of how well each object lies within its cluster (From Wiki).
Note that: This function is a rewriting version of the function "silhouette()" in R package cluster. The original function "silhouette()" is to compute the silhouette information based on a dissimilarity matrix. Here the silhouette_SimilarityMatrix() is to solve the computation based on the similarity matrix. The result of the silhouette_SimilarityMatrix() is compatible to the function "Silhouette()".

Usage

```
silhouette_SimilarityMatrix(group, similarity_matrix)
```

Arguments

`group` A vector represent the cluster label for a set of samples.

`similarity_matrix` A similarity matrix between samples

Details

For each observation i, the return sil[i,] contains the cluster to which i belongs as well as the neighbor cluster of i (the cluster, not containing i, for which the average dissimilarity between its observations and i is minimal), and the silhouette width s(i) of the observation.

Value

An object, sil, of class silhouette which is an [n x 3] matrix with attributes. The colnames correspondingly are c("cluster", "neighbor", "sil_width").

Author(s)

Xu,Taosheng <taosheng.x@gmail.com>,Thuc Le <Thuc.Le@unisa.edu.au>

References

Rousseeuw, P.J. (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math., 20, 53-65.

See Also

silhouette

Examples

```
data(GeneExp)
data(miRNAExp)
GBM=list(GeneExp=GeneExp,miRNAExp=miRNAExp)
result=ExecuteSNF(GBM, clusterNum=3, K=20, alpha=0.5, t=20)
sil=silhouette_SimilarityMatrix(result$group, result$distanceMatrix)
plot(sil)
###If use the silhouette(), the result is wrong because the input is a similarity matrix.
sil1=silhouette(result$group, result$distanceMatrix)
plot(sil1)  ##wrong result
```

`spectralAlg` *This is an internal function but need to be exported for the function ExecuteSNF.CC() call.*

Description

This is Spectral Clustering Algorithm extracted from SNFtools package spectralClustering() with a tiny modification.

Usage

```
spectralAlg(affinity, K, type = 3)
```

Arguments

`affinity`	Similarity matrix
`K`	Number of clusters
`type`	The variants of spectral clustering to use.

Value

A vector consisting of cluster labels of each sample.

Examples

```
####see the spectralClustering() in SNFtool package for the detail example.
data(miRNAExp)
Dist1=SNFtool::dist2(t(miRNAExp),t(miRNAExp))
W1 = SNFtool::affinityMatrix(Dist1, 20, 0.5)
group=spectralAlg(W1,3, type = 3)
```

`status` *Dataset: Survival status*

Description

- A vector representing the survival status for GBM cancer patients matched with the "GeneExp" and "miRNAExp" . 0=alive or censored, 1=dead

Format

A numeric vector

Examples

```
data(status)
```

`survAnalysis` *Survival analysis(Survival curves, Log-rank test) and compute Silhouette information for cancer subtypes*

Description

Survival analysis is a very common tool to explain and validate the cancer subtype identification result. It provides the significance testing and graphical display for the verification of the survival patterns between the identified cancer subtypes.

Usage

```
survAnalysis(
  mainTitle = "Survival Analysis",
  time,
  status,
  group,
  distanceMatrix = NULL,
  similarity = TRUE
)
```

Arguments

`mainTitle`	A character will display in the result plot.
`time`	A numeric vector representing the survival time (days) of a set of samples.
`status`	A numeric vector representing the survival status of a set of samples. 0=alive/censored, 1=dead.
`group`	A vector represent the cluster label for a set of samples.
`distanceMatrix`	A data matrix represents the similarity matrix or dissimilarity matrix between samples. If NULL, it will not compute silhouette width and draw the plot.
`similarity`	A logical value. If TRUE, the distanceMatrix is a similarity distance matrix between samples. Otherwise a dissimilarity distance matrix between samples

Value

The log-rank test p-value

Author(s)

Xu,Taosheng <`taosheng.x@gmail.com`>,Thuc Le <`Thuc.Le@unisa.edu.au`>

Examples

```
data(GeneExp)
data(miRNAExp)
data(time)
data(status)
data1=FSbyCox(GeneExp,time,status,cutoff=0.05)
data2=FSbyCox(miRNAExp,time,status,cutoff=0.05)
GBM=list(GeneExp=data1,miRNAExp=data2)
```

```
### SNF result analysis
result1=ExecuteSNF(GBM, clusterNum=3, K=20, alpha=0.5, t=20)
group1=result1$group
distanceMatrix1=result1$distanceMatrix
p_value1=survAnalysis(mainTitle="GBM_SNF",time,status,group1,
                     distanceMatrix=distanceMatrix1,similarity=TRUE)

### WSNF result analysis
data(Ranking)
####Retrieve there feature ranking for genes
gene_Name=rownames(data1)
index1=match(gene_Name,Ranking$mRNA_TF_miRNA.v21_SYMBOL)
gene_ranking=data.frame(gene_Name,Ranking[index1,],stringsAsFactors=FALSE)
index2=which(is.na(gene_ranking$ranking_default))
gene_ranking$ranking_default[index2]=min(gene_ranking$ranking_default,na.rm =TRUE)
####Retrieve there feature ranking for genes
miRNA_ID=rownames(data2)
index3=match(miRNA_ID,Ranking$mRNA_TF_miRNA_ID)
miRNA_ranking=data.frame(miRNA_ID,Ranking[index3,],stringsAsFactors=FALSE)
index4=which(is.na(miRNA_ranking$ranking_default))
miRNA_ranking$ranking_default[index4]=min(miRNA_ranking$ranking_default,na.rm =TRUE)
###Clustering
ranking1=list(gene_ranking$ranking_default ,miRNA_ranking$ranking_default)
result2=ExecuteWSNF(datasets=GBM, feature_ranking=ranking1, beta = 0.8, clusterNum=3,
                    K = 20,alpha = 0.5, t = 20, plot = TRUE)
group2=result2$group
distanceMatrix2=result2$distanceMatrix
p_value2=survAnalysis(mainTitle="GBM_WSNF",time,status,group2,
                     distanceMatrix=distanceMatrix2,similarity=TRUE)
```

TCGA500 *Dataset: TCGA500 ExpressionSet*

Description

The TCGA breast cancer datasets is downloaded from the TCGA data portal and the datasets consist of level 3 mRNA and miRNA, lncRNA expression data from multiple platforms.

Format

A ExpressionSet

Details

- ExpressionSet

Examples

```
data(TCGA500)
```

`time` *Dataset: Survival time*

Description

- A vector representing the right censored survival time (days) for GBM cancer patients matched with the "GeneExp" and "miRNAExp" datasets.

Format

A numeric vector

Examples

```
data(time)
```

Index