Package 'STATegRa'

May 2, 2026

Type Package

Title Classes and methods for multi-omics data integration

Version 1.49.0

Date 2018-05-02

Author STATegra Consortia

Maintainer David Gomez-Cabrero <david.gomezcabrero@ki.se>, Núria Planell <nuria.planell.picola@navarra.es>

Depends R (>= 2.10)

Imports Biobase, gridExtra, ggplot2, methods, stats, grid, MASS, calibrate, gplots, edgeR, limma, foreach, affy

Suggests RUnit, BiocGenerics, knitr (>= 1.6), rmarkdown, BiocStyle (>= 1.3), roxygen2, doSNOW

VignetteBuilder knitr

Encoding UTF-8

biocViews Software, StatisticalMethod, Clustering, DimensionReduction, PrincipalComponent

Description Classes and tools for multi-omics data integration.

License GPL-2

LazyLoad yes

Collate 'STATegRa_combiningMappings.R' 'STATegRa_fused.R' 'STATegRa_holistOmics.R' 'STATegRa_omicsCLUST_bioMap.R' 'STATegRa_omicsCLUST_bioDist.R' 'STATegRa_omicsCLUST_bioDistW.R' 'STATegRa_omicsNPC.R' 'STATegRa_omicsNPC_internal.R' 'STATegRa_omicsPCA_caClass.R' 'STATegRa_omicsPCA_methods.R' 'STATegRa_omicsPCA_plotting.R' 'STATegRa_package.R'

RoxygenNote 5.0.1

git_url https://git.bioconductor.org/packages/STATegRa

git_branch devel

git_last_commit 384a5d7

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Contents

`bioDist` *bioDist*

Description

Function to compute a bioDistclass object from profile data and a mapping. For details of the process see the user's guide, but briefly the process involves using the mapping to identify reference features appropriate to each surrogate feature (if any), aggregating the surrogate data into pseudo-data for each reference feature, and then calculating the correlation distance between the reference features according to the surrogate data.

Usage

```
bioDist(referenceFeatures=NULL, reference=NULL, mapping=NULL,
                referenceData=NULL, surrogateData=NULL, filtering=NULL,
                noMappingDist=NA, distance="spearman", aggregation="sum",
                maxitems=NULL, selectionRule="maxFC", expfac=NULL,
                name=NULL, ...)
```

Arguments

referenceFeatures	subset of features to be considered for the computation of the distances. If NULL then the features are first gathered from the features in referenceData. If referenceData is not provided then the list of features are gathered from mapping (bioMap class) and using the reference.
reference	A character indicating the variable that is being used as features to compute distance between
mapping	The mapping between feature types
referenceData	ExpressionSet object with the data from the reference features.
surrogateData	ExpressionSet object with the data from the surrogate features.
filtering	A filtering for the bioMap class. To be implemented.
noMappingDist	Distance value to be used when a reference feature do not map to any surrogate feature. If "max", maximum indirect distance among the rest of reference features is taken. If NA, distance weights are re-scaled so this surrogate association is not considered. If a number then the missing values are replaces with that value.
distance	Distance between features to be computed. Possible values are "pearson", "kendall", "spearman", "euclidean", "maximum", "manhattan", "canberra", "binary" and "minkowski". Default is "spearman".
aggregation	Action to perform when a reference feature maps to more than one surrogate feature. Options are "max", "sum", "mean" or "median" and the the values are aggregated according to the chosen statistic.
maxitems	The maximum number of surrogate features per reference feature to be used, selected according to "selectionRule" parameter. Default is 2.
selectionRule	Rule to select the surrogate features to be used (the number is determined by "maxitems"). It can be one of the following: (1) "maxcor" those presenting maximum correlation with corresponding main feature; in this case "referenceData" must be provided and the columns must overlap in at least 3 samples; (2) "maxmean": average across samples is computed and those features with higher mean are selected; case (3) is simmilar to (2) but considering other statistics: "maxmedian", "maxdiff", "maxFC", "sd" , "ee".
expfac	Not in use yet.
name	Character that describes the nature of the bioDist class computed
...	extra arguments passed to dist, eg "p=value" for the power used if calculating minkowski distance

Value

An object of class bioDistclass containing distances between the features in surrogateData.

Author(s)

David Gomez-Cabrero

Examples

```
data(STATegRa_S1)
data(STATegRa_S2)
require(Biobase)

# Truncate data for brevity
Block1 <- Block1[1:100,]
Block2 <- Block2[1:100,]

## Create ExpressionSets
mRNA.ds <- createOmicsExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicsExpressionSet(Data=Block2,pData=ed,pDataDescr=c("classname"))

## Create the bioMap
map.gene.miRNA<-bioMap(name = "Symbol-miRNA",
                       metadata =  list(type_v1="Gene",type_v2="miRNA",
                                        source_database="targetscan.Hs.eg.db",
                                        data_extraction="July2014"),
                       map=mapdata)

# Create Gene-gene distance computed through miRNA data
bioDistmiRNA<-bioDist(referenceFeatures = rownames(Block1),
                      reference = "Var1",
                      mapping = map.gene.miRNA,
                      surrogateData = miRNA.ds,  ### miRNA data
                      referenceData = mRNA.ds,  ### mRNA data
                      maxitems=2,
                      selectionRule="sd",
                      expfac=NULL,
                      aggregation = "sum",
                      distance = "spearman",
                      noMappingDist = 0,
                      filtering = NULL,
                      name = "mRNAbymiRNA")

# Create Gene-gene distance through mRNA data
bioDistmRNA<-new("bioDistclass",
                 name = "mRNAbymRNA",
                 distance = cor(t(exprs(mRNA.ds)),method="spearman"),
                 map.name = "id",
                 map.metadata = list(),
                 params = list())

###### Generation of the list of Surrogated distances.

bioDistList<-list(bioDistmRNA,bioDistmiRNA)
sample.weights<-matrix(0,4,2)
sample.weights[,1]<-c(0,0.33,0.67,1)
sample.weights[,2]<-c(1,0.67,0.33,0)

###### Generation of the list of bioDistWclass objects.

bioDistWList<-bioDistW(referenceFeatures = rownames(Block1),
                       bioDistList = bioDistList,
                       weights=sample.weights)
```

```
###### Plot of distances.
bioDistWPlot(referenceFeatures = rownames(Block1) ,
             listDistW = bioDistWList,
             method.cor="spearman")

###### Computing the matrix of features/distances associated.

fm<-bioDistFeature(Feature = rownames(Block1)[1] ,
                   listDistW = bioDistWList,
                   threshold.cor=0.7)
bioDistFeaturePlot(data=fm)
```

`bioDistclass` *bioDistclass*

Description

Class to manage mappings between genomic features.

Usage

```
bioDistclass(name, distance, map.name, map.metadata, params)
```

Arguments

`name`	Name assigned to the object
`distance`	Matrix giving the distance between features
`map.name`	Charactering giving the name of the bioMap object used to compute the distance
`map.metadata`	List of parameters used to generate the mapping
`params`	List of parameters used to generate the distance

`bioDistFeature` *bioDistFeature*

Description

Function that computes for a given selected feature the closest features given a selected set of weighted distances.

Usage

```
bioDistFeature(Feature, listDistW, threshold.cor)
```

Arguments

`Feature`	Feature A selected as a reference.
`listDistW`	A list of bioDistWclass objects. All the objects must contain the Feature A selected and all of them must contain the same set of features.
`threshold.cor`	A threshold to select the features associated to Feature A

Value

Matrix with the associated features given the different weighted distances considered

Author(s)

David Gomez-Cabrero

Examples

```
data(STATegRa_S1)
data(STATegRa_S2)
require(Biobase)

# Truncate data for brevity
Block1 <- Block1[1:100,]
Block2 <- Block2[1:100,]

## Create ExpressionSets
mRNA.ds <- createOmicsExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicsExpressionSet(Data=Block2,pData=ed,pDataDescr=c("classname"))

## Create the bioMap
map.gene.miRNA<-bioMap(name = "Symbol-miRNA",
                       metadata =  list(type_v1="Gene",type_v2="miRNA",
                                        source_database="targetscan.Hs.eg.db",
                                        data_extraction="July2014"),
                       map=mapdata)

# Create Gene-gene distance computed through miRNA data
bioDistmiRNA<-bioDist(referenceFeatures = rownames(Block1),
                      reference = "Var1",
                      mapping = map.gene.miRNA,
                      surrogateData = miRNA.ds,  ### miRNA data
                      referenceData = mRNA.ds,  ### mRNA data
                      maxitems=2,
                      selectionRule="sd",
                      expfac=NULL,
                      aggregation = "sum",
                      distance = "spearman",
                      noMappingDist = 0,
                      filtering = NULL,
                      name = "mRNAbymiRNA")

# Create Gene-gene distance through mRNA data
bioDistmRNA<-new("bioDistclass",
                 name = "mRNAbymRNA",
                 distance = cor(t(exprs(mRNA.ds)),method="spearman"),
                 map.name = "id",
                 map.metadata = list(),
                 params = list())

###### Generation of the list of Surrogated distances.

bioDistList<-list(bioDistmRNA,bioDistmiRNA)
sample.weights<-matrix(0,4,2)
sample.weights[,1]<-c(0,0.33,0.67,1)
```

```
sample.weights[,2]<-c(1,0.67,0.33,0)

###### Generation of the list of bioDistWclass objects.

bioDistWList<-bioDistW(referenceFeatures = rownames(Block1),
                       bioDistList = bioDistList,
                       weights=sample.weights)

###### Plot of distances.
bioDistWPlot(referenceFeatures = rownames(Block1) ,
             listDistW = bioDistWList,
             method.cor="spearman")

###### Computing the matrix of features/distances associated.

fm<-bioDistFeature(Feature = rownames(Block1)[1] ,
                   listDistW = bioDistWList,
                   threshold.cor=0.7)
bioDistFeaturePlot(data=fm)
```

`bioDistFeaturePlot` *bioDistFeaturePlot*

Description

Function that pltos the results from a bioDistFeature analysis

Usage

```
bioDistFeaturePlot(data)
```

Arguments

`data`	Matrix produced by `bioDistFeature`

Value

Generates a heatmap plot

Author(s)

David Gomez-Cabrero

Examples

```
data(STATegRa_S1)
data(STATegRa_S2)
require(Biobase)

# Truncate data for brevity
Block1 <- Block1[1:100,]
Block2 <- Block2[1:100,]
```

```
## Create ExpressionSets
mRNA.ds <- createOmicsExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicsExpressionSet(Data=Block2,pData=ed,pDataDescr=c("classname"))

## Create the bioMap
map.gene.miRNA<-bioMap(name = "Symbol-miRNA",
                       metadata =  list(type_v1="Gene",type_v2="miRNA",
                                        source_database="targetscan.Hs.eg.db",
                                        data_extraction="July2014"),
                       map=mapdata)

# Create Gene-gene distance computed through miRNA data
bioDistmiRNA<-bioDist(referenceFeatures = rownames(Block1),
                      reference = "Var1",
                      mapping = map.gene.miRNA,
                      surrogateData = miRNA.ds,  ### miRNA data
                      referenceData = mRNA.ds,  ### mRNA data
                      maxitems=2,
                      selectionRule="sd",
                      expfac=NULL,
                      aggregation = "sum",
                      distance = "spearman",
                      noMappingDist = 0,
                      filtering = NULL,
                      name = "mRNAbymiRNA")

# Create Gene-gene distance through mRNA data
bioDistmRNA<-new("bioDistclass",
                 name = "mRNAbymRNA",
                 distance = cor(t(exprs(mRNA.ds)),method="spearman"),
                 map.name = "id",
                 map.metadata = list(),
                 params = list())

###### Generation of the list of Surrogated distances.

bioDistList<-list(bioDistmRNA,bioDistmiRNA)
sample.weights<-matrix(0,4,2)
sample.weights[,1]<-c(0,0.33,0.67,1)
sample.weights[,2]<-c(1,0.67,0.33,0)

###### Generation of the list of bioDistWclass objects.

bioDistWList<-bioDistW(referenceFeatures = rownames(Block1),
                       bioDistList = bioDistList,
                       weights=sample.weights)

###### Plot of distances.
bioDistWPlot(referenceFeatures = rownames(Block1) ,
             listDistW = bioDistWList,
             method.cor="spearman")

###### Computing the matrix of features/distances associated.

fm<-bioDistFeature(Feature = rownames(Block1)[1] ,
                   listDistW = bioDistWList,
                   threshold.cor=0.7)
```

```
bioDistFeaturePlot(data=fm)
```

---

bioDistW *bioDistW*

Description

Function that computes weighted distances between a list of bioDistclass objects.

Usage

```
bioDistW(referenceFeatures, bioDistList, weights)
```

Arguments

referenceFeatures	The set of features that weighted distance is computed between.
bioDistList	A list of bioDistclass objects. All the objects must contain the set of features selected.
weights	A matrix where the number of columns equals the number of elements included in the bioDistList list.

Value

Returns a list of bioDistWclass objects. Each element in the list returns the weighted distance associated to each row in the "weights" matrix.

Author(s)

David Gomez-Cabrero

Examples

```
data(STATegRa_S1)
data(STATegRa_S2)
require(Biobase)

# Truncate data for brevity
Block1 <- Block1[1:100,]
Block2 <- Block2[1:100,]

## Create ExpressionSets
mRNA.ds <- createOmicsExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicsExpressionSet(Data=Block2,pData=ed,pDataDescr=c("classname"))

## Create the bioMap
map.gene.miRNA<-bioMap(name = "Symbol-miRNA",
                       metadata =  list(type_v1="Gene",type_v2="miRNA",
                                        source_database="targetscan.Hs.eg.db",
                                        data_extraction="July2014"),
                       map=mapdata)
```

```
# Create Gene-gene distance computed through miRNA data
bioDistmiRNA<-bioDist(referenceFeatures = rownames(Block1),
                      reference = "Var1",
                      mapping = map.gene.miRNA,
                      surrogateData = miRNA.ds,  ### miRNA data
                      referenceData = mRNA.ds,  ### mRNA data
                      maxitems=2,
                      selectionRule="sd",
                      expfac=NULL,
                      aggregation = "sum",
                      distance = "spearman",
                      noMappingDist = 0,
                      filtering = NULL,
                      name = "mRNAbymiRNA")

# Create Gene-gene distance through mRNA data
bioDistmRNA<-new("bioDistclass",
                 name = "mRNAbymRNA",
                 distance = cor(t(exprs(mRNA.ds)),method="spearman"),
                 map.name = "id",
                 map.metadata = list(),
                 params = list())

###### Generation of the list of Surrogated distances.

bioDistList<-list(bioDistmRNA,bioDistmiRNA)
sample.weights<-matrix(0,4,2)
sample.weights[,1]<-c(0,0.33,0.67,1)
sample.weights[,2]<-c(1,0.67,0.33,0)

###### Generation of the list of bioDistWclass objects.

bioDistWList<-bioDistW(referenceFeatures = rownames(Block1),
                       bioDistList = bioDistList,
                       weights=sample.weights)

###### Plot of distances.
bioDistWPlot(referenceFeatures = rownames(Block1) ,
             listDistW = bioDistWList,
             method.cor="spearman")

###### Computing the matrix of features/distances associated.

fm<-bioDistFeature(Feature = rownames(Block1)[1] ,
                   listDistW = bioDistWList,
                   threshold.cor=0.7)
bioDistFeaturePlot(data=fm)
```

`bioDistWPlot` *bioDistWPlot*

Description

Function that plots the "distance relation" between features computed through different surrogate features.

Usage

```
bioDistWPlot(referenceFeatures, listDistW, method.cor)
```

Arguments

referenceFeatures	The set of features to be used.
listDistW	A list of bioDistWclass objects.
method.cor	Method to compute distances between the elements in the listDistW. The default is spearman correlation.

Value

Makes a plot with the projected distance between the listDistW objects.

Author(s)

David Gomez-Cabrero

Examples

```
data(STATegRa_S1)
data(STATegRa_S2)
require(Biobase)

# Truncate data for brevity
Block1 <- Block1[1:100,]
Block2 <- Block2[1:100,]

## Create ExpressionSets
mRNA.ds <- createOmicsExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicsExpressionSet(Data=Block2,pData=ed,pDataDescr=c("classname"))

## Create the bioMap
map.gene.miRNA<-bioMap(name = "Symbol-miRNA",
                       metadata =  list(type_v1="Gene",type_v2="miRNA",
                                        source_database="targetscan.Hs.eg.db",
                                        data_extraction="July2014"),
                       map=mapdata)

# Create Gene-gene distance computed through miRNA data
bioDistmiRNA<-bioDist(referenceFeatures = rownames(Block1),
                      reference = "Var1",
                      mapping = map.gene.miRNA,
                      surrogateData = miRNA.ds,  ### miRNA data
                      referenceData = mRNA.ds,  ### mRNA data
                      maxitems=2,
                      selectionRule="sd",
                      expfac=NULL,
                      aggregation = "sum",
                      distance = "spearman",
                      noMappingDist = 0,
                      filtering = NULL,
                      name = "mRNAbymiRNA")
```

```
# Create Gene-gene distance through mRNA data
bioDistmRNA<-new("bioDistclass",
                 name = "mRNAbymRNA",
                 distance = cor(t(exprs(mRNA.ds)),method="spearman"),
                 map.name = "id",
                 map.metadata = list(),
                 params = list())

###### Generation of the list of Surrogated distances.

bioDistList<-list(bioDistmRNA,bioDistmiRNA)
sample.weights<-matrix(0,4,2)
sample.weights[,1]<-c(0,0.33,0.67,1)
sample.weights[,2]<-c(1,0.67,0.33,0)

###### Generation of the list of bioDistWclass objects.

bioDistWList<-bioDistW(referenceFeatures = rownames(Block1),
                       bioDistList = bioDistList,
                       weights=sample.weights)

###### Plot of distances.
bioDistWPlot(referenceFeatures = rownames(Block1) ,
             listDistW = bioDistWList,
             method.cor="spearman")

###### Computing the matrix of features/distances associated.

fm<-bioDistFeature(Feature = rownames(Block1)[1] ,
                   listDistW = bioDistWList,
                   threshold.cor=0.7)
bioDistFeaturePlot(data=fm)
```

---

bioMap *bioMap*

Description

Function to generate a bioMap object.

Usage

```
bioMap(name, metadata, map)
```

Arguments

name	Name to assign the object
metadata	A list with information of the mapping. Elements expected in the list are: (1) "type_v1" and "type_v2", refer to the nature of the features mapped; a vocabulary we recommend is "gene", "mRNA", "miRNA", "proteins", etc. (2) "source_database", provides information on the source of the mapping; from a specific data-base e.g. "targetscan.Hs.eg.db" to a genomic location mapping. (3) "data_extraction" stores information on the data the mapping was generated or downloaded.

map A data.frame object storing the mapping. The data.frame may inclue an unlimited number of columns, however the first column must be named "Var1" and refer to the elements of "type_v1" and simmilarly for the second column ("Var2", "type_v2").

Value

An object of class bioMap

Author(s)

David Gomez-Cabrero

Examples

```
data(STATegRa_S2)
map.gene.miRNA<-bioMap(name = "Symbol-miRNA",
                       metadata = list(type_v1="Gene",type_v2="miRNA",
                                       source_database="targetscan.Hs.eg.db",
                                       data_extraction="July2014"),
                       map=mapdata)
```

---

caClass-class *caClass*

Description

Stores the results of any of the omicsPCA analyses.

Slots

InitialData List of ExpressionSets, one for each set of omics data

Names Character vector giving names for the input data

preprocessing Character vector describing the preprocessing applied to the data

preproData List of matrices containing data after preprocessing

caMethod Character giving the component analysis method name

commonComps Numeric giving the number of common components

distComps Numeric vector giving the number of distinctive components for each block

scores List of matrices of common and distinctive scores

loadings List of matrices of common and distinctive loadings

VAF List of matrices indicating VAF (Variability Explained For) for each component in each block of data

others List containing other miscellaneous information specific to different SCA methods

Author(s)

Patricia Sebastian Leon

combiningMappings *combiningMappings, combining several mappings for use in the omicsNPC function*

Description

This function combines several annotation so that measurements across different datasets are mapped to the same reference elements (e.g., genes). The annotations should all be either data frame / matrices, named vectors/lists, or bioMap objects. See the examples for further details

Usage

```
combiningMappings(mappings, reference = NULL, retainAll = FALSE)
```

Arguments

mappings	List of annotations.
reference	If the annotations are data frame, matrices or bioMap objects, the name of the column containing the reference elements
retainAll	Logical, if set to TRUE measurements that have no counterparts in other datasets are retained

Value

A data frame encoding the mapping across several dataset

Author(s)

Vincenzo Lagani

References

Nestoras Karathanasis, Ioannis Tsamardinos and Vincenzo Lagani. omicsNPC: applying the Non-Parametric Combination methodology to the integrative analysis of heterogeneous omics data. Submitted to PlosONE.

Examples

```
#Example 1
#Mapping with data frames
mRNA <- data.frame(gene = rep(c('G1', 'G2', 'G3'), each = 2), probeset = paste('p', 1:6, sep = ''));
methylation <- data.frame(gene = c(rep('G1', 3), rep('G2', 4)),
                                 methy = paste('methy', 1:7, sep = ''));
miRNA <- data.frame(gene = c(rep('G1', 2), rep('G2', 1), rep('G3', 2)),
                               miR = c('miR1', 'miR2', 'miR1', 'miR1', 'miR2'));
mappings <- list(mRNA = mRNA, methylation = methylation, miRNA = miRNA);
combiningMappings(mappings = mappings, retainAll = TRUE)

#Example 2
#Mapping with character vectors
mRNA <- rep(c('G1', 'G2', 'G3'), each = 2);
names(mRNA) = paste('p', 1:6, sep = '');
```

```
methylation <- c(rep('G1', 3), rep('G2', 4));
names(methylation) = paste('methy', 1:7, sep = '');
miRNA <- c(rep('G1', 2), rep('G2', 1), rep('G3', 2));
names(miRNA) = c('miR1', 'miR2', 'miR1', 'miR1', 'miR2');
mappings <- list(mRNA = mRNA, methylation = methylation, miRNA = miRNA);
combiningMappings(mappings = mappings, retainAll = TRUE)
```

`createOmicsExpressionSet` *createOmicsExpressionSet*

Description

This function allow to the user to create a ExpressionSet object from a matrix representing an omics dataset. It allows to include the experimental design and annotation in the ExpressionSet object.

Usage

```
createOmicsExpressionSet(Data, pData = NULL, pDataDescr = NULL,
  feaData = NULL, feaDataDescr = NULL)
```

Arguments

`Data`	Omics data
`pData`	Data associated with the samples/phenotype
`pDataDescr`	Description of the phenotypic data
`feaData`	Data associated with the variables/features annotation
`feaDataDescr`	Description of the feature annotation

Details

In Data matrix, samples has to be in columns and variables has to be in rows

Value

ExpressionSet with the data provided

Author(s)

Patricia Sebastian-Leon

Examples

```
data(STATegRa_S3)
B1 <- createOmicsExpressionSet(Data=Block1.PCA,pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,pData=ed.PCA,
                               pDataDescr=c("classname"))
```

`getInitialData` *Retrieve initial data from caClass objects*

Description

Generic function to retrieve the initial data used for by `omicsCompAnalysis` from a `caClass-class` object

Usage

```
getInitialData(x, block=NULL)
```

Arguments

`x`	`caClass-class` object.
`block`	Character indicating the block of data to be returned. It can be specified by the position of the block ("1" or "2") or the name assigned in the `caClass-class` object. If it is NULL both blocks are displayed.

Value

The requested data block or blocks

Author(s)

Patricia Sebastian-Leon

See Also

`omicsCompAnalysis`, `caClass-class`

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA, pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA, pDataDescr=c("classname"))
# Omics components analysis
res <- omicsCompAnalysis(Input=list(B1, B2), Names=c("expr", "mirna"),
                         method="DISCOSCA", Rcommon=2, Rspecific=c(2, 2),
                         center=TRUE, scale=TRUE, weight=TRUE)
getInitialData(res)
getInitialData(res, block="expr")
```

getLoadings *Retrieve component analysis loadings*

Description

Generic function to retrieve loadings (common and distinctive) found by omicsCompAnalysis on a caClass-class object.

Usage

```
getLoadings(x, part=NULL, block=NULL)
```

Arguments

x	caClass-class object.
part	Character indicating whether "common" or "distinctive" loadings should be displayed
block	Character indicating the block of data for which the loadings will be given. It can be specified by the position of the block ("1" or "2") or the name assigned in the caClass-class object. If it is NULL both blocks are displayed.

Value

A list containing the requested information.

Author(s)

Patricia Sebastian-Leon

See Also

omicsCompAnalysis, caClass-class

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA, pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA, pDataDescr=c("classname"))
# Omics components analysis
res <- omicsCompAnalysis(Input=list(B1, B2), Names=c("expr", "mirna"),
                         method="DISCOSCA", Rcommon=2, Rspecific=c(2, 2),
                         center=TRUE, scale=TRUE, weight=TRUE)
getLoadings(res)
getLoadings(res, part="common", block="expr")
getLoadings(res, part="distinctive", block="expr")
```

getMethodInfo *Retrieve information about component analysis method*

Description

Generic function to retrieve information about the method used by omicsCompAnalysis on a caClass-class object.

Usage

```
getMethodInfo(x, method=FALSE, comps=NULL, block=NULL)
```

Arguments

x	caClass-class object.
method	Logical indicating whether to return the method name.
comps	Character indicating which component number to return ("common", "distinctive" or "all")
block	Character indicating the block of data for which the component count will be given. It can be specified by the position of the block ("1" or "2") or the name assigned in the caClass-class object. If it is NULL both blocks are displayed.

Value

A list containing the requested information.

Author(s)

Patricia Sebastian-Leon

See Also

omicsCompAnalysis, caClass-class

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA, pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA, pDataDescr=c("classname"))
# Omics components analysis
res <- omicsCompAnalysis(Input=list(B1, B2), Names=c("expr", "mirna"),
                         method="DISCOSCA", Rcommon=2, Rspecific=c(2, 2),
                         center=TRUE, scale=TRUE, weight=TRUE)
getMethodInfo(res)
getMethodInfo(res, method=TRUE)
getMethodInfo(res, comps="all", block="expr")
```

getPreprocessing *Retrieve information about preprocessing*

Description

Generic function to retrieve information about the preprocessing done by omicsCompAnalysis on a caClass-class object.

Usage

```
getPreprocessing(x, process=FALSE, preproData=FALSE, block=NULL)
```

Arguments

x	caClass-class object.
process	Logical indicating whether to return information about the processing done.
preproData	Logical indicating whether to return the pre-processed data matrices.
block	Character indicating the block of data to be returned. It can be specified by the position of the block ("1" or "2") or the name assigned in the caClass-class object. If it is NULL both blocks are displayed.

Value

If both process and preproData are specified, a list containing (otherwise the individual item):

process Character indicating the processing done

preproData Matrix (or list of matrices, depending on block) containing pre-processed data

Author(s)

Patricia Sebastian-Leon

See Also

omicsCompAnalysis, caClass-class

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA, pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA, pDataDescr=c("classname"))
# Omics components analysis
res <- omicsCompAnalysis(Input=list(B1, B2), Names=c("expr", "mirna"),
                         method="DISCOSCA", Rcommon=2, Rspecific=c(2, 2),
                         center=TRUE, scale=TRUE, weight=TRUE)
getPreprocessing(res, process=TRUE)
getPreprocessing(res, preproData=TRUE, block="1")
```

getScores *Retrieve component analysis scores*

Description

Generic function to retrieve scores (common and distinctive) found by omicsCompAnalysis on a caClass-class object.

Usage

```
getScores(x, part=NULL, block=NULL)
```

Arguments

x	caClass-class object.
part	Character indicating whether "common" or "distinctive" scores should be displayed
block	Character indicating the block of data for which the scores will be given. It can be specified by the position of the block ("1" or "2") or the name assigned in the caClass-class object. If it is NULL both blocks are displayed.

Value

A list containing the requested information.

Author(s)

Patricia Sebastian-Leon

See Also

omicsCompAnalysis, caClass-class

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA, pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA, pDataDescr=c("classname"))
# Omics components analysis
res <- omicsCompAnalysis(Input=list(B1, B2), Names=c("expr", "mirna"),
                         method="DISCOSCA", Rcommon=2, Rspecific=c(2, 2),
                         center=TRUE, scale=TRUE, weight=TRUE)
getScores(res)
getScores(res, part="common")
getScores(res, part="distinctive", block="expr")
```

getVAF *Retrieve information abotut VAF*

Description

Generic function to retrieve VAF found by omicsCompAnalysis on a caClass-class object.

Usage

```
getVAF(x, part=NULL, block=NULL)
```

Arguments

x	caClass-class object.
part	Character indicating whether "common" or "distinctive" VAF should be displayed
block	Character indicating the block of data for which the VAF will be given. It can be specified by the position of the block ("1" or "2") or the name assigned in the caClass-class object. If it is NULL both blocks are displayed.

Value

A list containing the requested information.

Author(s)

Patricia Sebastian-Leon

See Also

omicsCompAnalysis, caClass-class

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA, pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA, pDataDescr=c("classname"))
# Omics components analysis
res <- omicsCompAnalysis(Input=list(B1, B2), Names=c("expr", "mirna"),
                         method="DISCOSCA", Rcommon=2, Rspecific=c(2, 2),
                         center=TRUE, scale=TRUE, weight=TRUE)
getVAF(res)
getVAF(res, part="common")
getVAF(res, part="distinctive", block="expr")
```

holistOmics *HolistOmics an application of NPC on omics datasets*

Description

This function is defunct. Use omicsNPC instead.

Usage

```
holistOmics(dataInput, dataTypes, comb.method = c("Fisher", "Liptak", "Tippett"),
               numPerm = 1000, numCores = 1, verbose = FALSE)
```

Arguments

dataInput	List of ExpressionSet objects, one for each data modality.
dataTypes	Character vector with possible values: 'RNA-seq', 'microarray'
comb.method	Character vector with possible values: 'Fisher', 'Liptak', 'Tippett', if more than one is specified, all will be used.
numPerm	Number of permutations
numCores	Number of CPU cores to use
verbose	Logical, if set to TRUE holistOmics prints out the step that it performs

Value

A data.frame

Author(s)

Nestoras Karathanasis

References

Pesarin, Fortunato, and Luigi Salmaso. Permutation tests for complex data: theory, applications and software. John Wiley & Sons, 2010.

Examples

```
# Load the data
data("TCGA_BRCA_Batch_93")
# Setting dataTypes, the first two ExpressionSets include RNAseq data,
# the third ExpressionSet includes Microarray data.
dataTypes <- c("RNAseq", "RNAseq", "Microarray")
# Setting methods to combine pvalues
comb.method = c("Fisher", "Liptak", "Tippett")
# Setting number of permutations
numPerm = 1000
# Setting number of cores
numCores = 1
# Setting holistOmics to print out the steps that it performs.
verbose = TRUE
# Run holistOmics analysis.
```

```
# The output is a data.frame of p-values.
# Each row corresponds to a gene name. Each column corresponds to a method
# used in the analysis.
## Not run: out <- holistOmics(dataInput = TCGA_BRCA_Data, dataTypes = dataTypes,
                            comb.method = comb.method, numPerm = numPerm,
                            numCores = numCores, verbose = verbose)
## End(Not run)
```

`modelSelection` *Find optimal common and distinctive components*

Description

Estimate the optimal number of common and distinctive components according to given selection criteria.

Usage

```
modelSelection(Input,Rmax,fac.sel,varthreshold=NULL,nvar=NULL,PCnum=NULL,center=FALSE,scale=FALS
```

Arguments

`Input`	List of `ExpressionSet` objects, one for each block of data
`Rmax`	Maximum common components
`fac.sel`	PCA criteria for selection ("%accum", "single%", "rel.abs", "fixed.num")
`varthreshold`	Cumulative variance criteria for PCA selection. Threshold for "%accum" or "single%" criteria.
`nvar`	Relative variance criteria. Threshold for "rel.abs".
`PCnum`	Fixed number of components for "fixed.num".
`center`	Character (or FALSE) specifying which (if any) centering will be applied before analysis. Choices are "PERBLOCKS" (each block separately) or "ALLBLOCKS" (all data together).
`scale`	Character (or FALSE) specifying which (if any) scaling will be applied before analysis. Choices are "PERBLOCKS" (each block separately) or "ALLBLOCKS" (all data together).
`weight`	Logical indicating whether weighting is to be done. Choices are "BETWEENBLOCKS"
`plot_common`	Logical indicating whether to plot the explained variances (SSQ) of each block and its estimation and the ratios
`plot_dist`	Logical indicating whether to plot the explained variances (SSQ) and the accumulated variance for each block

Value

List containing:

common List with common components results

commonComps Optimal number of common components

ssqs Matrix of SSQ for each block and estimator

pssq ggplot object showing SSQ for each block and estimator

pratios ggplot object showing SSQ ratios between each block and estimator

dist List containg the results of distinct PCA for each input block; for each block PCAres and numComps is returned within a list

PCAres List containing results of PCA, with fields "eigen", "var.exp", "scores" and "loadings"

nomComps Number of components selected

Author(s)

Patricia Sebastian-Leon

See Also

omicsCompAnalysis

Examples

```
data(STATegRa_S3)
B1 <- createOmicsExpressionSet(Data=Block1.PCA,pData=ed.PCA,pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,pData=ed.PCA,pDataDescr=c("classname"))
ms <- modelSelection(Input=list(B1, B2), Rmax=3, fac.sel="single\%", varthreshold=0.03, center=TRUE, scale=FA
ms
```

---

omicsCompAnalysis *Components analysis for multiple objects*

Description

This function performs a components analysis of object wise omics data to understand the mechanisms that underlay all the data blocks under study (common mechanisms) and the mechanisms underlying each of the data block independently (distinctive mechanisms). This analysis include both, the preprocessing of data and the components analysis by using three different methodologies.

Usage

```
omicsCompAnalysis(Input, Names, method, Rcommon, Rspecific,
                          convThres=1e-10, maxIter=600, center=FALSE,
                          scale=FALSE, weight=FALSE)
```

Arguments

Input	List of ExpressionSet objects, one for each block of data.
Names	Character vector giving names for each Input object.
method	Method to use for analysis (either "DISCOSCA", "JIVE", or "O2PLS").
Rcommon	Number of common components between all blocks
Rspecific	Vector giving number of unique components for each input block
convThres	Stop criteria for convergence
maxIter	Maximum number of iterations

center	Character (or FALSE) specifying which (if any) centering will be applied before analysis. Choices are "PERBLOCKS" (each block separately) or "ALLBLOCKS" (all data together).
scale	Character (or FALSE) specifying which (if any) scaling will be applied before analysis. Choices are "PERBLOCKS" (each block separately) or "ALLBLOCKS" (all data together).
weight	Logical indicating whether weighting is to be done.

Value

An object of class caClass-class.

Author(s)

Patricia Sebastian Leon

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA,pData=ed.PCA,
pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA,pDataDescr=c("classname"))
# Omics components analysis
discoRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                              method="DISCOSCA",Rcommon=2,Rspecific=c(2,2),
                              center=TRUE,scale=TRUE,weight=TRUE)
jiveRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                             method="JIVE",Rcommon=2,Rspecific=c(2,2),
                             center=TRUE,scale=TRUE,weight=TRUE)
o2plsRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                              method="O2PLS",Rcommon=2,Rspecific=c(2,2),
                              center=TRUE,scale=TRUE,weight=TRUE)
```

---

omicsNPC *omicsNPC, applying the Non-Parametric Combination (NPC) on omics datasets*

Description

This function applies the NonParametric Combination methodology on the integrative analysis of different omics data modalities. It retrieves genes associated to a given outcome, taking into account all omics data. First, each datatype is analyzed independently using the appropriate method. omicsNPC analyses continuous data (microarray) using limma, while count data (e.g., RNAseq) are first preprocessed with using the "voom" function. The user can also specify their own function for computing deregulation / association The p-values from the single dataset analysis are then combined employing Fisher, Liptak and Tippett combining functions. The Tippett function returns findings which are supported by at least one omics modality. The Liptak function returns findings which are supportd by most modalities. The Fisher function has an intermediate behavior between those of Tippett and Liptak.

Usage

```
omicsNPC(dataInput, dataMapping, dataTypes = rep('continuous', length(dataInput)),
                combMethods = c("Fisher", "Liptak", "Tippett"), numPerms = 1000,
                numCores = 1, verbose = FALSE, functionGeneratingIndex = NULL,
                outcomeName = NULL, allCombinations = FALSE,
                dataWeights = rep(1, length(dataInput))/length(dataInput),
                returnPermPvalues = FALSE, ...)
```

Arguments

dataInput	List of ExpressionSet objects, one for each data modality.
dataMapping	A data frame describing how to map measurements across datasets. See details for more information.
dataTypes	Character vector with possible values: 'continuous', 'count'. Alternatively, a list of functions for assessing deregulation / association with an outcome
combMethods	Character vector with possible values: 'Fisher', 'Liptak', 'Tippett'. If more than one is specified, all will be used.
numPerms	Number of permutations
numCores	Number of CPU cores to use
verbose	Logical, if set to TRUE omicsNPC prints out the step that it performs
functionGeneratingIndex	Function generating the indices for randomly permuting the samples
outcomeName	Name of the outcome of interest / experimental factor, as reported in the design matrices. If NULL, the last column of the design matrices is assumed to be the outcome of interest.
allCombinations	Logical, if TRUE all combinations of omics datasets are considered
dataWeights	A vector specifying the weigth to give to each dataset. Note that sum(dataWeights) should be 1.
returnPermPvalues	Logical, should the p-values computed at each permutation being returned?
...	Additional arguments to be passed to the user-defined functions

Value

A list containing: stats0 Partial deregulation / association statistics pvalues0 The partial p-values computed on each dataset pvaluesNPC The p-values computed through NPC. permPvalues The p-values computed at each permutation

Author(s)

Nestoras Karathanasis, Vincenzo Lagani

References

Pesarin, Fortunato, and Luigi Salmaso. Permutation tests for complex data: theory, applications and software. John Wiley & Sons, 2010. Nestoras Karathanasis, Ioannis Tsamardinos and Vincenzo Lagani. omicsNPC: applying the Non-Parametric Combination methodology to the integrative analysis of heterogeneous omics data. PlosONE 11(11): e0165545. doi:10.1371/journal.pone.0165545

Examples

```
# Load the data
data("TCGA_BRCA_Batch_93")
# Setting dataTypes, the first two ExpressionSets include RNAseq data,
# the third ExpressionSet includes Microarray data.
dataTypes <- c("count", "count", "continuous")
# Setting methods to combine pvalues
combMethods = c("Fisher", "Liptak", "Tippett")
# Setting number of permutations
numPerms = 1000
# Setting number of cores
numCores = 1
# Setting omicsNPC to print out the steps that it performs.
verbose = TRUE
# Run omicsNPC analysis.
# The output contains a data.frame of p-values, where each row corresponds to a gene,
# and each column corresponds to a method used in the analysis.

## Not run: out <- omicsNPC(dataInput = TCGA_BRCA_Data, dataTypes = dataTypes,
                            combMethods = combMethods, numPerms = numPerms,
                            numCores = numCores, verbose = verbose)
## End(Not run)
```

---

plotRes *Plot component analysis results*

Description

Plot scatterplots of scores or loadings, for common and distinctive parts as well as combined plots.

Usage

```
plotRes(object, comps=c(1, 2), what, type, combined, block=NULL,
          color=NULL, shape=NULL, labels=NULL, title=NULL, xlabel=NULL, ylabel=NULL, background=TR
               palette=NULL, pointSize=4, labelSize=NULL,
          axisSize=NULL, titleSize=NULL, sizeValues = c(2,4), shapeValues = c(17, 0))
```

Arguments

object	caClass-class containing component analysis results
comps	If combined=FALSE, it indicates the x and y components of the type and block chosen. If combined=TRUE, it indicates the component to plot for the first block of information and the component for the second block of information to plot together. By default the components are set to c(1,2) if combined=FALSE and to c(1,1) if combined=TRUE.
what	Either "scores", "loadings" or "both"
type	Either "common", "individual" or "both"
combined	Logical indicating whether to make a simple plot of two components from one block, or components from different blocks
block	Which block to plot, either "1" or "2" or the name of the block.
color	Character specifying a pData column from the original data to use to color points

shape	Character specifying a pData column to select point shape
labels	Character specifying a pData column from which to take point labels
title	Main title
xlabel	x-axis name
ylabel	y-axis name
background	Logical specifying whether to make a grey background
palette	Vector giving the color palette for the plot
pointSize	Size of plot points
labelSize	Size of point labels if not NULL
axisSize	Size of axis text
titleSize	Size of title text
sizeValues	Vector containing sizes for scores and loadings
shapeValues	Vector indicating the shapes for scores and loadings

Value

ggplot object

Author(s)

Patricia Sebastian-Leon

Examples

```
data("STATegRa_S3")
B1 <- createOmicsExpressionSet(Data=Block1.PCA,pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA,pDataDescr=c("classname"))
# Omics components analysis
discoRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                              method="DISCOSCA",Rcommon=2,Rspecific=c(2,2),
                              center=TRUE,scale=TRUE,weight=TRUE)
jiveRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                             method="JIVE",Rcommon=2,Rspecific=c(2,2),
                             center=TRUE,scale=TRUE,weight=TRUE)

o2plsRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                              method="O2PLS",Rcommon=2,Rspecific=c(2,2),
                              center=TRUE,scale=TRUE,weight=TRUE)

# Scatterplot of scores variables associated to common components

# DISCO-SCA
plotRes(object=discoRes,comps=c(1,2),what="scores",type="common",
        combined=FALSE,block=NULL,color="classname",shape=NULL,labels=NULL,
        background=TRUE,palette=NULL,pointSize=4,labelSize=NULL,
        axisSize=NULL,titleSize=NULL)
# JIVE
plotRes(object=jiveRes,comps=c(1,2),what="scores",type="common",
       combined=FALSE,block=NULL,color="classname",shape=NULL,labels=NULL,
        background=TRUE,palette=NULL,pointSize=4,labelSize=NULL,
```

```
        axisSize=NULL,titleSize=NULL)

# O2PLS
# Scatterplot of scores variables associated to common components
# Associated to first block
p1 <- plotRes(object=o2plsRes,comps=c(1,2),what="scores",type="common",
              combined=FALSE,block="expr",color="classname",shape=NULL,
              labels=NULL,background=TRUE,palette=NULL,pointSize=4,
              labelSize=NULL,axisSize=NULL,titleSize=NULL)
# Associated to second block
p2 <- plotRes(object=o2plsRes,comps=c(1,2),what="scores",type="common",
              combined=FALSE,block="mirna",color="classname",shape=NULL,
              labels=NULL,background=TRUE,palette=NULL,pointSize=4,
              labelSize=NULL,axisSize=NULL,titleSize=NULL)

# Combined plot of scores variables assocaited to common components
plotRes(object=o2plsRes,comps=c(1,1),what="scores",type="common",
        combined=TRUE,block=NULL,color="classname",shape=NULL,
        labels=NULL,background=TRUE,palette=NULL,pointSize=4,
        labelSize=NULL,axisSize=NULL,titleSize=NULL)

# Loadings plot for individual components
# Separately for each block
p1 <- plotRes(object=discoRes,comps=c(1,2),what="loadings",type="individual",
              combined=FALSE,block="expr",color="classname",shape=NULL,
              labels=NULL,background=TRUE,palette=NULL,pointSize=4,
              labelSize=NULL,axisSize=NULL,titleSize=NULL)
p2 <- plotRes(object=discoRes,comps=c(1,2),what="loadings",type="individual",
              combined=FALSE,block="mirna",color="classname",shape=NULL,
              labels=NULL,background=TRUE,palette=NULL,pointSize=4,
              labelSize=NULL,axisSize=NULL,titleSize=NULL)

# Biplot: scores + loadings
plotRes(object=discoRes,comps=c(1,2),what="both",type="common",
        combined=FALSE,block="expr",color="classname",shape=NULL,
        labels=NULL,background=TRUE,palette=NULL,pointSize=4,
        labelSize=NULL,axisSize=NULL,titleSize=NULL)
```

`plotVAF` *Plot VAF (Variance Explained For) from Component Analysis*

Description

This function visualises the VAF results from component analysis. The input is a `caClass-class` object from `omicsCompAnalysis`. VAF cannot be calculated if mode "O2PLS" was used. The plots for modes "DISCOSCA" and "JIVE" are different since DISCO-SCA distinctive components have some VAF in the other block. This VAF can be interpreted as an error in the rotation.

Usage

```
plotVAF(object, mainTitle="")
```

Arguments

object	caClass-class object containing component analysis results
mainTitle	Plot title

Value

ggplot object

Author(s)

Patricia Sebastian-Leon

Examples

```
data("STATegRa_S3")
require(ggplot2)
B1 <- createOmicsExpressionSet(Data=Block1.PCA,pData=ed.PCA,
                               pDataDescr=c("classname"))
B2 <- createOmicsExpressionSet(Data=Block2.PCA,
                               pData=ed.PCA,pDataDescr=c("classname"))
# Omics components analysis
discoRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                              method="DISCOSCA",Rcommon=2,Rspecific=c(2,2),
                              center=TRUE,scale=TRUE,weight=TRUE)
jiveRes <- omicsCompAnalysis(Input=list(B1,B2),Names=c("expr","mirna"),
                             method="JIVE",Rcommon=2,Rspecific=c(2,2),
                             center=TRUE,scale=TRUE,weight=TRUE)

# DISCO-SCA plotVAF
plotVAF(discoRes)

# JIVE plotVAF
plotVAF(jiveRes)
```

STATegRa *STATegRa*

Description

STATegRa is a package for the integrative analysis of multi-omic data-sets.

For full information, see the user's guide.

See Also

STATegRaUsersGuide

STATegRa-defunct *Defunct functions in STATegRa*

Description

These functions have are defunct and no longer available

Details

- holistOmics: replaced by omicsNPC

STATegRaUsersGuide *STATegRaUsersGuide*

Description

Finds the location of the STATegRa User's Guide and optionally opens it.

Usage

```
STATegRaUsersGuide(view = TRUE)
```

Arguments

view	Whether to open a browser

Value

The path to the documentation

Author(s)

David Gomez-Cabrero

Examples

```
STATegRaUsersGuide(view=FALSE)
```

STATegRa_data *STATegRa data*

Description

mRNA data (Block1), miRNA data (Block2) and the design matrix (ed), from STATegRa_S1, provides selected data downloaded from https://tcga-data.nci.nih.gov/docs/publications/gbm_exp/. The mapping between miRNA and mRNA (mapdata, available in STATegRa_S2) contains, as a processed matrix, selected information available from TargetScan; we selected the set of miRNA target predictions for humans for those miRNA-mRNA pairs where both miRNA and mRNA were in Block1 and Block2 respectively.

The PCA version of the data (Block1.PCA, Block2.PCA, ed.PCA; available in STATegRa_S3), provides a similar data-set to Block1, Block2 and ed data; however in this case the data has been processed in order to provide a pedagogic example of OmicsPCA. Results obtained from OmicsPCA (omicsCompAnalysis) with the existing data should not be taken as clinically valid.

Format

Two matrices with mRNA and miRNA expression data, a design matrix that describes both and a mapping between miRNA and genes.

Author(s)

David Gomez-Cabrero, Patricia Sebastian-Leon, Gordon Ball

Source

(a) See https://tcga-data.nci.nih.gov/docs/publications/gbm_exp/. (b) Gabor Csardi, targetscan.Hs.eg.db: TargetScan miRNA target predictions for human. R package version 0.6.1

Examples

```
data(STATegRa_S1)
data(STATegRa_S2)
data(STATegRa_S3)
```

STATegRa_data_TCGA_BRCA *STATegRa data*

Description

Data were downloaded from TCGA data portal, https://tcga-data.nci.nih.gov/tcga/. We downloaded sixteen tumour samples and the sixteen matching normal, for Breast invasive carcinoma, BRCA, batch 93. Herein, three types of data modalities are included, RNAseq (TCGA_BRCA_Data$RNAseq), RNAseqV2 (TCGA_BRCA_Data$RNAseqV2) and Expression-Genes (TCGA_BRCA_Data$Microarray). The Data Level was set to Level 3. For each data type, we pooled all data to one matrix, where rows corresponded to genes and columns to samples. Only the first 100 genes are included.

Format

One list, which contains three ExpressionSet objects.

Author(s)

Nestoras Karathanasis, Vincenzo Lagani

Source

See https://tcga-data.nci.nih.gov/tcga/.

Examples

```
# load data
data(TCGA_BRCA_Batch_93)
```

Index