Package 'STATegRa'

April 10, 2015

Type Package

Title Classes and methods for multi-omics data integration

Version 1.0.0

Date 2014-09-09

Author STATegra Consortia

Maintainer David Gomez-Cabrero <david.gomezcabrero@ki.se>, Patricia Sebastián-León <psebastian@cipf.es>, Gordon Ball <gordon.ball@ki.se>

Depends R (>= 2.10)

Imports Biobase, gridExtra, ggplot2, methods, grid, MASS, calibrate, gplots

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

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_fused.R STATegRa_omicsCLUST_bioMap.R STATegRa_omicsCLUST_bioDist.R STATegRa_omicsCLUST_bioDistW.R STATegRa_omicsPCA_caClass.R STATegRa_omicsPCA_methods.R STATegRa_omicsPCA_plotting.R

R topics documented:

`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

See Also

`bioDistclass`,`bioDistclass-constructor`

Examples

```
# Definition of a bioDistclass Object
data(STATegRa_S1)
data(STATegRa_S2)

# 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 bioDistclass Object
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")
```

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	A matrix object that stores the distance between features.
map.name	Character that stores the name of the bioMap object used to compute the distance.
map.metadata	a list of parameters used to generate the mapping
params	a list of parameters used to generate the distance

Value

An object of class bioDistclass

Author(s)

David Gomez-Cabrero

See Also

bioMap

Examples

```
# Definition of a bioDistclass Object
data(STATegRa_S1)
## Create ExpressionSets
mRNA.ds <- createOmicsExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))

require(Biobase)

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

bioDistclass-class *bioDistclass*

Description

Class to manage and store surrogates distances between features.

Constructor

bioDistclass

Author(s)

David Gomez-Cabrero

See Also

bioDistclass-constructor

Examples

```
# Definition of a bioDistclass Object
data(STATegRa_S1)
## Create ExpressionSets
mRNA.ds <- createOmicsExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))

require(Biobase)

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

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

Returns a matrix with the associated features to feature A given the different weighted distances considered.

Author(s)

David Gomez-Cabrero

See Also

bioDistclass,bioDistclass-constructor

Examples

```
# Definition of a bioDistclass Object
data(STATegRa_S1)
data(STATegRa_S2)

# 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)

require(Biobase)

# 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)

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

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

---

bioDistFeaturePlot *bioDistFeaturePlot*

Description

Function that pltos the results from a bioDistFeature analysis.

Usage

```
bioDistFeaturePlot(data)
```

Arguments

data	A matrix, the output of a bioDistFeature analysis

Value

Generates a plot with the computed summary.

Author(s)

David Gomez-Cabrero

See Also

bioDistFeature,bioDistW

Examples

```
# Definition of a bioDistclass Object
data(STATegRa_S1)
data(STATegRa_S2)

# 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)

require(Biobase)

# 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)

######  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

See Also

bioDistclass,bioDistclass-constructor

Examples

```
# Definition of a bioDistclass Object
data(STATegRa_S1)
data(STATegRa_S2)

# 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)

require(Biobase)

# 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)
```

`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 distances between bioDistWclass objects.

Author(s)

David Gomez-Cabrero

See Also

`bioDistW`

Examples

```
# Definition of a bioDistclass Object
data(STATegRa_S1)
data(STATegRa_S2)

# 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)

require(Biobase)

# 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")
```

`bioMap-class` *bioMap class*

Description

Class to manage mappings between omic features or identities.

Constructor

bioMap

Author(s)

David Gomez-Cabrero

See Also

bioMap-constructor

Examples

```
# Definition of a bioMap Object

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)
```

bioMap-constructor *bioMap constructor*

Description

Function to generate a bioMap class object.

Usage

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

Arguments

name	name assigned to 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

See Also

bioMap-class

Examples

```
# Definition of a bioMap Object

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)
```

biplotRes *Biplot results of Components Analysis*

Description

Function to display a biplot of an caClass object coming from a Component Analysis.

Usage

```
biplotRes(object,type,comps,block,title=NULL,colorCol=NULL,sizeValues=c(2,4),
          shapeValues=c(17,0),background=TRUE,pointSize=4,labelSize=NULL,
          axisSize=NULL,titleSize=NULL)
```

Arguments

object	caClass object containing results of Components Analysis
type	Indicate which components should be represented, "common", "individual" or "both" in the case of combined plots
comps	Components to plot. 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.
block	Indicates which block has to be represented. It can be specify by a numeric value (1 or 2) or a character (name of the block in the input data provided to the omicCompAnalysis() analysis)
title	Title of the biplot
colorCol	Character indicating with column of the pData in the initial ExpressionSet has to be used to set color in the graph. By default is the first column of the pData
sizeValues	Vector indicating which sizes has to be use for scores and loadings?
shapeValues	Vector indicating which shapes has to be used for scores and loadings
background	Logical indicating if the plot have to be represented with grey background or not
pointSize	Size of the points represented in the plot
labelSize	Size of the labels if label is not NULL
axisSize	Size of the text in the axis of the plot
titleSize	Size of main title

Value

Plot of class ggplot

Author(s)

Patricia Sebastian-Leon

See Also

ggplot,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"))
# 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)

# Biplot common part. DISCO-SCA

biplotRes(object=discoRes,type="common",comps=c(1,2),block="",
          title=NULL,colorCol="classname",sizeValues=c(2,4),
          shapeValues=c(17,0),background=TRUE,pointSize=4,
          labelSize=NULL,axisSize=NULL,titleSize=NULL)

# Biplot common part. O2PLS

p1 <- biplotRes(object=o2plsRes,type="common",comps=c(1,2),
          block="expr",title=NULL,colorCol="classname",
          sizeValues=c(2,4),shapeValues=c(17,0),
          background=TRUE,pointSize=4,labelSize=NULL,
          axisSize=NULL,titleSize=NULL)
p2 <- biplotRes(object=o2plsRes,type="common",comps=c(1,2),
          block="mirna",title=NULL,colorCol="classname",
          sizeValues=c(2,4),shapeValues=c(17,0),
          background=TRUE,pointSize=4,labelSize=NULL,
          axisSize=NULL,titleSize=NULL)

# Biplot distinctive part. O2PLS

p1 <- biplotRes(object=discoRes,type="individual",comps=c(1,2),
          block="expr",title=NULL,colorCol="classname",
          sizeValues=c(2,4),shapeValues=c(17,0),
          background=TRUE,pointSize=4,labelSize=NULL,
          axisSize=NULL,titleSize=NULL)
p2 <- biplotRes(object=discoRes,type="individual",comps=c(1,2),
          block="mirna",title=NULL,colorCol="classname",
```

```
sizeValues=c(2,4),shapeValues=c(17,0),
background=TRUE,pointSize=4,labelSize=NULL,
axisSize=NULL,titleSize=NULL)
```

---

caClass-class *caClass*

Description

Stores the results of any of the omicsPCA analyses.

Author(s)

Patricia Sebastian Leon

See Also

omicsCompAnalysis Accessor methods: getInitialData, getMethodInfo, getScores, getLoadings, getVAF, getPreprocessing

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)
# all caClass instances
class(discoRes)
class(jiveRes)
class(o2plsRes)
```

createOmicsExpressionSet *Create an ExpressionSet object since omics dataset.*

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	Matrix with omics dataset.
pData	Matrix indicating the phenotype of the samples (columns). rownames(pData)==colnames(exprs)
pDataDescr	Description of the phenotypic data
feaData	Matrix indicating 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

A object of class ExpressionSet

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 in a components analysis omicsCompAnalysis from an caClass-class object

Usage

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

Arguments

x	caClass-class object. If only this parameter is specified all information about the preprocessing of the data, as well as the data after preprocessing is applied, is returned
block	Character indicating the block of data that has to be returned. It can be specified by using the position of the block ("1" or "2") or the name assigned to this block in the caClass-class object. If it is NULL both blocks are displayed

Value

It returns a list with the information indicated by the parameters. By default a list with the following components:

process	methods applied to preprocess the data
preproData	Preprocessed 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)
```

```
getInitialData(res)
getInitialData(res, block="expr")
```

`getLoadings` *Retrieve information about the loadings obtained using* `omicsCompAnalysis` *from an* `caClass-class`

Description

Generic function to retrieve loadings (common and distinctive) obtained in the `omicsCompAnalysis` from an `caClass-class`

Usage

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

Arguments

`x`	`caClass-class` object. If only this parameter is specified loadings for common and distinctive part are diplayed
`part`	Character indicating if loadings for "common" or "distinctive" part has to be displayed
`block`	Character indicating the block of data whose scores has to be returned. It can be specified by using the position of the block ("1" or "2") or the name assigned to this block in the `caClass-class` object. If it is NULL both loadings are displayed

Value

By default, a list with loadings associated to both parts (common and distinctive). If any parameter is specified, the selected information is displayed.

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 the components analysis done using* `omicsCompAnalysis` *from an* `caClass-class`

Description

Generic function to retrieve information about the method and the number of components (common and individual) employed in the `omicsCompAnalysis` from an `caClass-class`

Usage

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

Arguments

`x`	`caClass-class` object. If only this parameter is specified all information about the method and number of components is diplayed
`method`	Logical indicating if method employed has to be returned (FALSE by default)
`comps`	Character indicating which components number has to be returned (The options are: "common" for common components, "distinctive" for distinctive components
`block`	Character indicating the block of data whose number of components has to be returned. It can be specified by using the position of the block ("1" or "2") or the name assigned to this block in the `caClass-class` object. If it is NULL both number of distinctive components are displayed

Value

It returns a list with the information indicated by the parameters. By default a list with the following components:

`method`	The component analysis method used for the analysis
`comps`	Number of components (common and distinctive) used as input for the method

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 the preprocessing done in* omicsCompAnalysis *from an* caClass-class

---

Description

Generic function to retrieve information about preprocessing done in omicsCompAnalysis from an caClass-class

Usage

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

Arguments

x	caClass-class object. If only this parameter is specified loadings for common and distinctive part are diplayed
process	Logical indicating whether to return information about the processing done. Defaults to FALSE.
preproData	Logical indicating whether to return the pre-processed data matrices. Defaults to FALSE.
block	Character indicating the block of data whose pre-processed state has to be returned, if preproData=TRUE. It can be specified by using the position of the block ("1" or "2") or the name assigned to this block in the caClass-class object. If it is NULL both blocks are displayed

Value

If process and preproData are both requested, a list containing:

process	methods applied to preprocess the data
preproData	Preprocessed data (either a single block or a list of blocks)

Otherwise, either a character vector for process or either a matrix or list of matrices for preproData, depending on block.

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 information about the scores obtained using* omicsCompAnalysis *from an* caClass-class

---

Description

Generic function to retrieve scores (common and distinctive) obtained in the omicsCompAnalysis from an caClass-class

Usage

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

Arguments

x	caClass-class object. If only this parameter is specified scores for common and distinctive part are diplayed
part	Character indicating if scores for "common" or "distinctive" part has to be displayed
block	Character indicating the block of data whose scores has to be returned. It can be specified by using the position of the block ("1" or "2") or the name assigned to this block in the caClass-class object. If it is NULL both scores are displayed

Value

By default, a list with scores associated to both parts (common and distinctive). If any parameter is specified, the selected information is displayed.

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 about the VAF obtained using* omicsCompAnalysis *from an* caClass-class

---

Description

Generic function to retrieve loadings (common and distinctive) obtained in the omicsCompAnalysis from an caClass-class

Usage

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

Arguments

x	caClass-class object. If only this parameter is specified loadings for common and distinctive part are diplayed
part	Character indicating if VAF for "common" or "distinctive" part has to be displayed. If the method is DISCO-SCA, another option, "cross", indicating the associated error due to the rotation is allowed
block	Character indicating the block of data whose VAF has to be returned. It can be specified by using the position of the block ("1" or "2") or the name assigned to this block in the caClass-class object. If it is NULL both VAFs are displayed

Details

It is important to note that in the case of O2PLS method, it is not possible to calculate the VAF because the components are not orthogonal

Value

By default, a list with VAF associated to both parts (common and distinctive). If any parameter is specified, the selected information is displayed.

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")
```

`modelSelection` *Number of optimal common and distinctive components in object-wise data*

Description

This function combine the function `selectCommonComps` and `PCA.selection` to stimate the optimal number of common and distinctive components in object-wise data. That is, it calculates the optimal number of common and individual components depending on the maximal number of common components and the individual components selection criteria provided by the user. The result is a list with the optimal number of common and distinctive components. This optimal number can be use for component analysis with function `omicsCompAnalysis`

Usage

```
modelSelection(Input,Rmax,fac.sel,varthreshold=NULL,nvar=NULL,PCnum=NULL)
```

Arguments

`Input`	List of `ExpressionSet` objects, one for each block of data
`Rmax`	Maximum number of common components
`fac.sel`	Criterium for selecting number of components. The posible option are: "%accum", "single%", "rel.abs" and "fixed.num"
`varthreshold`	Threshold for the selection of components in "%accum", "single%" criterions
`nvar`	Threshold applied when the option "rel.abs" is selected
`PCnum`	Fixed number of components to select when the option "fixed.num" is selected

Value

The function returns a list with the following components:

`common`	Number of optimal common components
`dist`	Number of optimal distictive components for each block

Author(s)

Patricia Sebastian-Leon

See Also

`selectCommonComps`,`PCA.selection`,`omicsCompAnalysis`

Examples

```
data(STATegRa_S3)

## Create ExpressionSets
# Block1 - Expression data
B1 <- createOmicsExpressionSet(Data=Block1.PCA,pData=ed.PCA,pDataDescr=c("classname"))
# Block2 - miRNA expression data
B2 <- createOmicsExpressionSet(Data=Block2.PCA,pData=ed.PCA,pDataDescr=c("classname"))

## Model Selection
ms <- modelSelection(Input=list(B1,B2),Rmax=4,fac.sel="single%",
                     varthreshold=0.03)
ms
```

---

omicsCompAnalysis *Components analysis for the analysis of object wise data*

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=10^(-10),maxIter=600,center=FALSE,
                  scale=FALSE,weight=FALSE)
```

Arguments

Input	List of ExpressionSet objects, one for each block of data
Names	Vector indicating a distintive name for each block of data
method	Method used for components analysis. The methods included are DISCO_SCA, JIVE and O2PLS
Rcommon	Number of common components between all blocks
Rspecific	Vector indicating the individual components of each block of data
convThres	Maximum value for convergence (stop criterium. Default value: 0.0000001)
maxIter	Maximum number of iterations (stop criterium. Default value: 600)
center	Character indicating which type of centering must be applied to data before the analysis. Data can be centered considering all blocks together (PERBLOCKS) or each block separately (PERBLOCKS). If the parameter has a NULL (default) value, no center is applied

scale	Character indicating which type of scaling must be applied to data before the analysis. Data can be scaled considering all blocks together (PERBLOCKS) or each block separately (PERBLOCKS). If the parameter has a NULL (default) value, no scale is applied
weight	Logical indicating if the blocks has to be weighted before the analysis

Value

A object of class caClass-class

Author(s)

Patricia Sebastian Leon

See Also

caClass-class

Examples

```
# Loading data
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)
```

PCA.selection *Selection of optimal number of components using PCA*

Description

Function to select the number of optimal components using Principal Components Analysis. There are four different criterions to select the number of components: accumulated variance explained, the individual explained variance of each component, the absolute value of its variability or just a fixed number of components.

Usage

```
PCA.selection(Data,fac.sel,varthreshold=NULL,nvar=NULL,PCnum=NULL)
```

Arguments

Data	Matrix with data. Samples has to be in columns and variables in rows
fac.sel	Criterium for selecting number of components. The posible option are: "%accum", "single%", "rel.abs" and "fixed.num"
varthreshold	Threshold for the selection of components in "%accum", "single%" criterions
nvar	Threshold applied when the option "rel.abs" is selected
PCnum	Fixed number of components to select when the option"fixed.num" is selected

Value

The function returns a list with the following components:

PCAres	A list containing the results of the PCA decomposition
numComps	Number of selected components applying the selected criterion

The components of PCAres list are the following:

eigen	Eigen values of the decomposition
var.exp	Explained variance of each component
scores	Matrix of scores
loadings	Matrix of loadings

Author(s)

Patricia Sebastian Leon

See Also

modelSelection

Examples

```
data(STATegRa_S3)
ps <- PCA.selection(Data=Block2.PCA,fac.sel="single%",varthreshold=0.03)
ps$numComps
```

plotRes *Plot results of Components Analysis*

Description

Function that allows to plot scatterplots for scores or loadings, for common and distinctive part as well as combined plots for plotting both parts together. Different combinations of the parameters produce different plots.

Usage

```
plotRes(object, comps=c(1,2), what, type, combined, block, color=NULL,
        shape=NULL, labels=NULL, background=TRUE, palette=NULL, pointSize=4,
        labelSize=NULL, axisSize=NULL, titleSize=NULL)
```

Arguments

object	caClass object containing results of Components Analysis
comps	Components to plot. 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	what=c("scores","loadings"). Are scores or loadings had to be represented?
type	Indicate which components should be represented, "common", "individual" or "both" in the case of combined plots
combined	Logical indicating if the plot is a simple plot representing two components from the same block of information, or combined representing in the same plot two component each of them from a different block of information.
block	Indicates which block has to be represented. It can be specify by a numeric value (1 or 2) or a character (name of the block in the input data provided to the omicCompAnalysis() analysis)
color	Character indicating with column of the pData in the initial ExpressionSet has to be used to set color in the graph. By default is the first column of the pData
shape	Character indicating with column of the pData in the initial ExpressionSet has to be used to set shape in the graph.
labels	Character indicating with column of the pData in the initial ExpressionSet has to be used to set labels in the graph.
background	Logical indicating if the plot have to be represented with grey background or not
palette	Vector indicating the pallete of colors used for the plot. See ggplot
pointSize	Size of the points represented in the plot
labelSize	Size of the labels if label is not NULL
axisSize	Size of the text in the axis of the plot
titleSize	Size of main title

Value

Plot of class ggplot

Author(s)

Patricia Sebastian Leon

See Also

ggplot,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"))
# 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="",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="",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="",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)

# Combined plot
plotRes(object=discoRes,comps=c(1,1),what="loadings",type="individual",
        combined=TRUE,block="",color="classname",shape=NULL,
        labels=NULL,background=TRUE,palette=NULL,pointSize=4,
        labelSize=NULL,axisSize=NULL,titleSize=NULL)
```

`plotVAF` *Function to plot the VAF (Variance Explained For) obtained from Component Analysis.*

Description

This function allows to the user to visualize the VAF results coming from a component analysis. The input is an object resulting from `omicsCompAnalysis` function of class `caClass`. The variance explained for (VAF) each component is given in VAF slot, that is a list containing the VAF for common and distinctive components. In the case of O2PLS, VAF cannot be calculated, because components are not orthogonal. Structure of plots associated to DISCO-SCA and JIVE are different. In the case of DISCO-SCA, components of individual blocks have an associated error due to the rotation is not perfect. This is the reason because the DISCO-SCA distinctive components have VAF in the other block. This VAF not associated to the corresponding block could be interpreted as the error for not having a perfect rotation.

Usage

```
plotVAF(object,mainTitle)
```

Arguments

object	caClass object containing results of Components Analysis
mainTitle	Title of the plot

Value

Plot representing the VAF of common and distintive components of each block.

Author(s)

Patricia Sebastian -Leon

See Also

omicsCompAnalysis

Examples

```
# Loading data
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)
require(ggplot2)

# DISCO-SCA plotVAF
plotVAF(discoRes)

# JIVE plotVAF
plotVAF(jiveRes)
```

---

selectCommonComps *Select common components of two blocks of object wise omics data.*

Description

This function applies a Simultaneous Component Analysis (SCA). The idea is that the scores for both blocks should have a similar behavior if the components are in the common mode. For evaluate if a component is common or not, the ratios be- tween the explained variances (SSQ) of each block and its estimationare used. The highest component having its ratios between 0.8 and 1.5 is selected as the optical number of common components.

Usage

```
selectCommonComps(X,Y,Rmax)
```

Arguments

X	First block of omics data
Y	Second block of omics data
Rmax	Maximum number of common components

Value

The function returns a list with the following components:

common	Number of optimal common components
ssqs	Matrix with SSQ for each block and its estimation
pssq	ggplot object representing SSQ for each block and its estimation
pratios	ggplot object representing the SSQ ratios between each block and its estimation

Author(s)

Patricia Sebastian-Leon

See Also

modelSelection,ggplot

Examples

```
data(STATegRa_S3)

cc <- selectCommonComps(X=Block1.PCA,Y=Block2.PCA,Rmax=3)
cc$common
# Graphical output
cc$pssq
cc$pratios
```

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

Examples

```
## Not run: STATegRaUsersGuide()
```

STATegRaUsersGuide *View STATegRa User's Guide*

Description

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

Usage

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

Arguments

view	logical, if TRUE the document will be opened using the default PDF document reader?

Details

The function `vignette("STATegRa")` will find the short STATegRa Vignette which describes how to obtain the STATegRa User's Guide. The User's Guide is not itself a true vignette because it is not automatically generated using Sweave during the package build process. This means that it cannot be found using `vignette`, hence the need for this special function.

If the operating system is other than Windows, then the PDF viewer used is that given by `Sys.getenv("R_PDFVIEWER")`. The PDF viewer can be changed using `Sys.putenv(R_PDFVIEWER=)`.

Value

Character string giving the file location. If `view=TRUE`, the PDF document reader is started and the User's Guide is opened, as a side effect.

Author(s)

David Gomez-Cabrero, but modifying slightly the available code from Gordon Smyth in edgeR

See Also

system

Examples

```
# To get the location:
STATegRaUsersGuide(view=FALSE)
# To open in pdf viewer:
## Not run: STATegRaUsersGuide()
```

STATegRa_initial_data *STATegRa_initial_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; howere in thiscase 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.

Usage

```
STATegRa_S1
```

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 2014-10-03

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.

Index