

# Package ‘STATegRa’

March 21, 2016

**Type** Package

**Title** Classes and methods for multi-omics data integration

**Version** 1.4.0

**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, 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\_fused.R' 'STATegRa\_holistOmics.R'  
'STATegRa\_holistOmics\_internal.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'  
'STATegRa\_package.R'

**NeedsCompilation** no

## R topics documented:

bioDist . . . . . 2

bioDistclass . . . . .	5
bioDistFeature . . . . .	5
bioDistFeaturePlot . . . . .	7
bioDistW . . . . .	9
bioDistWPlot . . . . .	11
bioMap . . . . .	13
biplotRes . . . . .	14
caClass-class . . . . .	16
createOmicsExpressionSet . . . . .	16
getInitialData . . . . .	17
getLoadings . . . . .	18
getMethodInfo . . . . .	19
getPreprocessing . . . . .	20
getScores . . . . .	21
getVAF . . . . .	22
holistOmics . . . . .	23
modelSelection . . . . .	25
omicsCompAnalysis . . . . .	26
PCA.selection . . . . .	27
plotRes . . . . .	28
plotVAF . . . . .	30
selectCommonComps . . . . .	31
STATegRa . . . . .	32
STATegRaUsersGuide . . . . .	32
STATegRa_data . . . . .	33
STATegRa_data_TCGA_BRCA . . . . .	33
<b>Index</b>	<b>35</b>

---

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 simmlar 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 <code>dist</code> , 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 <- createOmicExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicExpressionSet(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	<i>bioDistclass</i>
--------------	---------------------

---

### 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	<i>bioDistFeature</i>
----------------	-----------------------

---

### 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 plots 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 <- createOmicExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicExpressionSet(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 <- createOmicExpressionSet(Data=Block1,pData=ed,pDataDescr=c("classname"))
miRNA.ds <- createOmicExpressionSet(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 include an unlimited number of columns, however the first column must be named "Var1" and refer to the elements of "type_v1" and similarly 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)
```

---

 biplotRes

*Biplot of component analysis*


---

**Description**

Generate a biplot of component analysis results

**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 component analysis results
type	Character specifying which components to plot; "common", "individual" or "both"
comps	Components to plot. If combined=FALSE, specifies the component indices to use as x and y for the plot. Otherwise, the component from the first block and the component from second block to plot together.
block	Which block to plot, either "1" or "2" or the name of the block.
title	Plot title
colorCol	Character specifying a pData column to use to colorise the plot points
sizeValues	Vector containing sizes for scores and loadings
shapeValues	Vector indicating the shapes for scores and loadings
background	Logical, whether to use a grey background
pointSize	Size of plot points
labelSize	Size of plot labels if not NULL
axisSize	Size of axis text
titleSize	Size of title text

**Value**

ggplot2 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)

# 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.

### 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

---

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	<i>Retrieve initial data from caClass objects</i>
----------------	---

---

**Description**

Generic function to retrieve the initial data used for by [omicsCompAnalysis](#) from a [caClass-class](#) object

**Usage**

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

**Arguments**

x	<a href="#">caClass-class</a> 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 <a href="#">caClass-class</a> 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	<a href="#">caClass-class</a> 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 <a href="#">caClass-class</a> 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	<a href="#">caClass-class</a> 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 <a href="#">caClass-class</a> 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	<a href="#">caClass-class</a> 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 <a href="#">caClass-class</a> 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	<a href="#">caClass-class</a> 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 <a href="#">caClass-class</a> 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 about VAF*

---

**Description**

Generic function to retrieve VAF found by [omicsCompAnalysis](#) on a [caClass-class](#) object.

**Usage**

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

**Arguments**

x	<a href="#">caClass-class</a> 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 <a href="#">caClass-class</a> 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="DISCOCA", 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 applies the NonParametric Combination methodology on the integrative analysis of different omics data modalities. It retrieves differential expressed genes between control and treatment, taking into account all omics data. First, each datatype is analyzed independently using the appropriate method. HolistOmics analyses static (one time point) RNAseq, using Voom+limma method, and static (one time point) Microarray, using limma. Then, the resulting p-values are combined employing Fisher, Liptak and Tippett combining functions. Tippett function returns findings which are supported by at least one omics modality. Liptak function returns findings which are supported by most modalities. Fisher function has an intermediate behavior between those of Tippett and Liptak.

**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	<i>Find optimal common and distinctive components</i>
----------------	---

---

### Description

Uses [selectCommonComps](#) and [PCA.selection](#) to 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)
```

### Arguments

Input	List of two ExpressionSet objects
Rmax	Maximum common components (see <a href="#">selectCommonComps</a> )
fac.sel	PCA criteria (see <a href="#">PCA.selection</a> )
varthreshold	Cumulative variance criteria (see <a href="#">PCA.selection</a> )
nvar	Relative variance criteria (see <a href="#">PCA.selection</a> )
PCnum	Fixed component number (see <a href="#">PCA.selection</a> )

### Value

List containing:

**common** Number of common components

**dist** Number of distinct components per input block

### Author(s)

Patricia Sebastian-Leon

### See Also

[selectCommonComps](#), [PCA.selection](#), [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=4, fac.sel="single%", varthreshold=0.03)
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)

```

PCA.selection

*Select an optimal number of components using PCA***Description**

Selects the optimal number of components from data using PCA. There are four different criteria available: accumulated variance explained, individual explained variance of each component, absolute value of variability or fixed number of components.

**Usage**

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

**Arguments**

Data	Data matrix (with samples in columns and features in rows)
fac.sel	Selection criteria ("%accum", "single%", "rel.abs", "fixed.num")
varthreshold	Threshold for "%accum" or "single%" criteria
nvar	Threshold for "rel.abs"
PCnum	Fixed number of components for "fixed.num"

**Value**

List containing:

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

**numComps** Number of components selected

**Author(s)**

Patricia Sebastian Leon

**Examples**

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

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,
        color=NULL, shape=NULL, labels=NULL, background=TRUE,
        palette=NULL, pointSize=4, labelSize=NULL,
        axisSize=NULL, titleSize=NULL)
```

**Arguments**

object	caClass object 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" or "loadings"
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
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

**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="",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 associated 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

*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

<code>object</code>	<code>caClass-class</code> object containing component analysis results
<code>mainTitle</code>	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)

```

---

selectCommonComps      *Select common components in two data blocks*

---

**Description**

This function applies a Simultaneous Component Analysis (SCA). The idea is that the scores for both blocks should have a similar behaviour if the components are in the common mode. Evaluation is by the ratios between the explained variances (SSQ) of each block and the estimator. The highest component count with  $0.8 < \text{ratio} < 1.5$  is selected.

**Usage**

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

**Arguments**

X	Matrix of omics data
Y	Matrix of omics data
Rmax	Maximum number of common components to find

**Value**

A list with components:

**common** Optimal number of common components

**ssqs** Matrix of SSQ for each block and estimator

**pssq** `ggplot` object showing SSQ for each block and estimator

**pratio** `ggplot` object showing SSQ ratios between each block and estimator

**Author(s)**

Patricia Sebastian-Leon

**Examples**

```
data(STATegRa_S3)
cc <- selectCommonComps(X=Block1.PCA, Y=Block2.PCA, Rmax=3)
cc$common
cc$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](#)

---

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/](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/](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. More information about these types of data as well as the "data level" can be found <https://tcga-data.nci.nih.gov/tcga/tcgaDataType.jsp> For each data type, we pooled all data to one matrix, where rows corresponded to genes and columns to samples. Then, columns and rows were aligned across matrices so as each gene and each sample to correspond to the same position. Here only the first 100 genes are included.

**Format**

One list, which contains three ExpressionSet objects.

**Author(s)**

Nestoras Karathanasis

**Source**

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

**Examples**

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

# Index

\*Topic **datagen**  
  createOmicsExpressionSet, 16

bioDist, 2  
bioDist, character, character, bioMap, ExpressionSet, ExpressionSet, ExpressionSet, 20  
  (bioDist), 2  
bioDistclass, 5  
bioDistFeature, 5  
bioDistFeature, character, list, numeric-method  
  (bioDistFeature), 5  
bioDistFeaturePlot, 7  
bioDistW, 9  
bioDistW, character, list, matrix-method  
  (bioDistW), 9  
bioDistWPlot, 11  
bioDistWPlot, character, list, character-method  
  (bioDistWPlot), 11  
bioMap, 13  
biplotRes, 14  
biplotRes, caClass, character, numeric, character-method  
  (biplotRes), 14  
Block1 (STATegRa\_data), 33  
Block2 (STATegRa\_data), 33

caClass-class, 16  
createOmicsExpressionSet, 16  
createOmicsExpressionSet, matrix-method  
  (createOmicsExpressionSet), 16

dist, 3

ed (STATegRa\_data), 33

getInitialData, 17  
getInitialData, caClass-method  
  (getInitialData), 17  
getLoadings, 18  
getLoadings, caClass-method  
  (getLoadings), 18  
getMethodInfo, 19  
getMethodInfo, caClass-method  
  (getMethodInfo), 19  
getPreprocessing, 20  
getPreprocessing, caClass-method  
  (getPreprocessing), 20  
getScores, 21  
getScores, caClass-method (getScores), 21  
getVAF, 22  
getVAF, caClass-method (getVAF), 22  
ggplot, 31

holistOmics, 23  
holistOmics, list, character-method  
  (holistOmics), 23

mapdata (STATegRa\_data), 33  
modelSelection, 25  
modelSelection, list, numeric, character-method  
  (modelSelection), 25

omicsCompAnalysis, 17–23, 25, 26, 30, 33  
omicsCompAnalysis, list, character, character, numeric, numeric-method  
  (omicsCompAnalysis), 26

PCA.selection, 25, 27  
PCA.selection, matrix, character-method  
  (PCA.selection), 27

plotRes, 28  
plotRes, caClass, numeric, character, character, logical, character-method  
  (plotRes), 28  
plotVAF, 30  
plotVAF, caClass-method (plotVAF), 30

selectCommonComps, 25, 31  
selectCommonComps, matrix, matrix, numeric-method  
  (selectCommonComps), 31

STATegRa, 32  
STATegRa-package (STATegRa), 32  
STATegRa\_data, 33  
STATegRa\_data\_TCGA\_BRCA, 33  
STATegRaUsersGuide, 32, 32

TCGA\_BRCA\_Data  
(STATegRa\_data\_TCGA\_BRCA), [33](#)