Package 'MineICA'

October 18, 2017

Type Package

Title Analysis of an ICA decomposition obtained on genomics data

Version 1.16.0

Date 2012-03-16

Author Anne Biton

Maintainer Anne Biton <anne.biton@gmail.com>

Description The goal of MineICA is to perform Independent Component Analysis (ICA) on multiple transcriptome datasets, integrating additional data (e.g molecular, clinical and pathological). This Integrative ICA helps the biological interpretation of the components by studying their association with variables (e.g sample annotations) and gene sets, and enables the comparison of components from different datasets using correlation-based graph.

License GPL-2

LazyLoad yes

Depends R (>= 2.10), methods, BiocGenerics (>= 0.13.8), Biobase, plyr, ggplot2, scales, foreach, xtable, biomaRt, gtools, GOstats, cluster, marray, mclust, RColorBrewer, colorspace, igraph, Rgraphviz, graph, annotate, Hmisc, fastICA, JADE

Imports AnnotationDbi, lumi, fpc, lumiHumanAll.db

Suggests biomaRt, GOstats, cluster, hgu133a.db, mclust, igraph, breastCancerMAINZ, breastCancerTRANSBIG, breastCancerUPP, breastCancerVDX

Enhances doMC

Collate 'AllClasses.R' 'AllGeneric.R' 'methods-IcaSet.R' 'methods-MineICAParams.R' 'compareAnalysis.R' 'functions_comp2annot.R' 'functions_comp2annottests.R' 'functions_enrich.R' 'functions.R' 'heatmap.plus.R' 'heatmapsOnSel.R' 'runAn.R' 'compareGenes.R'

biocViews Visualization, MultipleComparison

NeedsCompilation no

R topics documented:

A *Retrieve and set Source S and Mixing matrix A from IcaSet*

Description

These generic functions access and set the attributes `S`, `SByGene` and `A` stored in an object of class `IcaSet`.

Usage

```
S(object)
S(object) <- value
SByGene(object)
SByGene(object) <- value
A(object)
A(object) <- value
nbComp(object)
```

Arguments

`object`	object of class `IcaSet`
`value`	Data.frame with rows representing: features (for `S`), genes (for `SByGene`), or samples (for `A`) and columns representing components.

Value

`S` returns a data.frame containing feature projection values; `SByGene` returns a data.frame containing gene projection values; `A` returns a data.frame containing sample contribution values. `nbComp` returns the number of components, i.e the number of columns of `A`.

Author(s)

Anne Biton

Alist *Retrieve sample contributions stored in an IcaSet object as a list.*

Description

This generic function retrieves, from an IcaSet object, the sample contributions contained in the attribute A as a list where sample IDs are preserved.

Usage

```
Alist(object)
```

Arguments

object	Object of class IcaSet.

Value

Alist returns a list whose length equals the number of components contained in the IcaSet object. Each element of this list contains a vector of sample contributions indexed by the sample IDs.

Author(s)

Anne Biton

See Also

IcaSet-class

annot2Color *Association of a colour with each annotation level*

Description

Given a data.frame consisting of sample annotations, this function returns a vector which gives a colour per annotation level.

Usage

```
annot2Color(annot)
```

Arguments

annot	a data.frame containing the sample annotations (of dimension 'samples x annotations').

Details

Arbitrary colours are attributed to some specific annotations met by the author, and for the remaining annotation levels, the colours are attributed using packages RColorBrewer and rcolorspace.

Value

A vector of colours indexed by the annotation levels.

Author(s)

Anne Biton

---

annotCarbayo *Carbayo annotation data*

Description

Contains annotations for 93 samples of Carbayo data.

Author(s)

Anne Biton

References

http://jco.ascopubs.org/content/24/5/778/suppl/DC1

---

annotFeatures *Annotation of features using an annotation package*

Description

This function annotates a set of features

Usage

```
annotFeatures(features, type, annotation)
```

Arguments

features	Feature IDs to be annotated
type	The object from the package used to annotate the features, must be available in ls("package:package_name")
annotation	An annotation package

Value

A vector of gene/object IDs indexed by the feature IDs.

Author(s)

Anne Biton

Examples

```
library(hgu133a.db)
annotFeatures(features = c("1007_s_at", "1053_at", "117_at", "121_at", "1255_g_at"),
              type="SYMBOL", annotation="hgu133a.db")
```

annotFeaturesComp *Features annotation*

Description

##' This function annotates the features of an object of class IcaSet, and fills its attributes SByGene and datByGene.

Usage

```
annotFeaturesComp(icaSet, params,
  type = toupper(typeID(icaSet)["geneID_annotation"]),
  featureId = typeID(icaSet)["featureID_biomart"],
  geneId = typeID(icaSet)["geneID_biomart"])
```

Arguments

icaSet	An object of class IcaSet whose features have to be annotated. The attribute annotation of this object contains the annotation package to be used.
params	An object of class MineICAParams containing the parameters of the analysis.
type	The ID of the object of the annotation package to be used for the annotation, must be available in ls("package:package_name")
featureId	The type of the feature IDs, in the biomaRt way (type listFilters(mart) to choose one). Used when annotation(icaSet) is of length 0.
geneId	The type of the gene IDs, in the biomaRt way (type listAttributes(mart) to choose one). Used when annotation(icaSet) is of length 0.

Details

This function is called by function annotInGene which will check the validity of the attributes annotation, typeID, chipManu and eventually chipVersion of icaSet. If available, the attribute annotation of argument icaSet must be an annotation package and will be used to annotate the featureNames of icaSet. If attribute annotation of argument icaSet is not available (of length 0), biomaRt is used to annotate the features.

This function fills the attributes SByGene and datByGene of the argument icaSet. When several feature IDs are available for a same gene ID, the median value of the corresponding features IDs is attributed to the gene (the median of projection values is used for attribute SByGene, and the median of expression values is used for attribute datByGene).

When attribute chipManu of the argument icaSet is "illumina", the features are first converted into nuID using the package 'lumi*Mapping' and then annotated into genes. In that case, features can only be annotated in ENTREZID or SYMBOL. It means that typeID(icaSet)['geneID_annotation'] must be either 'ENTREZID' or 'SYMBOL'. You will need to annotate yourself the IcaSet object if you want to use different IDs.

Value

This function returns the argument icaSet with attributes SByGene and datByGene filled.

Author(s)

Anne Biton

See Also

annotFeatures, annotFeaturesWithBiomaRt, annotInGene

Examples

```
## load an example of IcaSet
data(icaSetCarbayo)
params <- buildMineICAParams()
require(hgu133a.db)
####=====================================================
## Use of annotation package contained in annotation(icaSet)
####=====================================================
## annotation in SYMBOL
icaSetCarbayo_annot <- annotFeaturesComp(icaSet=icaSetCarbayo, params=params, type="SYMBOL")
# arg 'type' is optional since the function uses contents of typeID(icaSet) as the defaults,
# it is specified in these examples for pedagogy views

## annotation in Entrez Gene
icaSetCarbayo_annot <- annotFeaturesComp(icaSet=icaSetCarbayo, params=params, type="ENTREZID")

## Not run:
####=====================================================
## Use of biomaRt, when annotation(icaSet) is of length 0
####=====================================================
## empty attribute 'annotation' of the IcaSet object
# when this attribute is not specified, biomaRt is used for annotation
annotation(icaSetCarbayo) <- character()

# make sure the mart attribute is correctly defined
mart(icaSetCarbayo) <- useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")

## make sure elements "featureID_biomaRt" and "geneID_biomaRt" of typeID(icaSet) are correctly filled
# they will be used by function 'annotFeaturesComp' through biomaRt to query the database
typeID(icaSetCarbayo)

## run annotation of HG-U133A probe set IDs into Gene Symbols using biomaRt
icaSetCarbayo_annot <- annotFeaturesComp(icaSet=icaSetCarbayo, params=params)


## End(Not run)
```

---

annotFeaturesWithBiomaRt

Annotation of features using biomaRt

Description

This function annotates a set of features using biomaRt

Usage

```
annotFeaturesWithBiomaRt(features, featureId, geneId,
  mart = useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl"))
```

Arguments

features	Feature IDs to be annotated
featureId	The type of the feature IDs, in the biomaRt way (type listFilters(mart) to choose one)
geneId	The type of the gene IDs, in the biomaRt way (type listAttributes(mart) to choose one)
mart	The mart object (database and dataset) used for annotation, see function useMart of package biomaRt

Value

A vector of gene IDs indexed by the feature IDs.

Author(s)

Anne Biton

Examples

```
if (interactive()) {
# define the database to be queried by biomaRt
mart <- useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")

# annotate a set of HG-U133a probe sets IDs into Gene Symbols
annotFeaturesWithBiomaRt(features = c("1007_s_at", "1053_at", "117_at", "121_at", "1255_g_at"),
featureId="affy_hg_u133a", geneId="hgnc_symbol", mart=mart)

# annotate a set of Ensembl Gene IDs into Gene Symbols
annotFeaturesWithBiomaRt(features = c("ENSG00000101412", "ENSG00000112242",
                                "ENSG00000148773", "ENSG00000131747", "ENSG00000170312",
                               "ENSG00000117399"), featureId="ensembl_gene_id", geneId="hgnc_symbol", mart=m
}
```

---

annotInGene *Features annotation of an object of class IcaSet.*

Description

This function annotates the features of an IcaSet object and fills its attributes SByGene and datByGene.

Usage

```
annotInGene(icaSet, params, annot = TRUE)
```

Arguments

icaSet	An object of class IcaSet to be annotated, must contain a valid annotation attribute.
params	An object of class MineICAParams containing the parameters of the analysis.
annot	TRUE (default) if the IcaSet object must indeed be annotated

Details

When attribute annotation of icaSet is not specified (of length 0), biomaRt is used to annotate the features through function annotFeaturesWithBiomaRt.

When specified, attribute annotation of argument icaSet must be an annotation package and will be used to annotate the featureNames of icaSet. In addition, the attribute typeID (a vector) of argument icaSet must contain a valid element geneID_annotation that determines the object of the package to be used for the annotation, see IcaSet.

When argument annot is TRUE, this function fills the attributes SByGene and datByGene of icaSet. When several feature IDs are available for a same gene ID, the median value of the corresponding features IDs is attributed to the gene (the median of the projection values is used for attribute SByGene, and the median of the expression values is used for attribute datByGene).

When attribute chipManu of the argument icaSet is "illumina", the features are first converted into nuID using the package 'lumi*Mapping' and then annotated into genes. In that case, features can only be annotated in ENTREZID or SYMBOL. It means that typeID(icaSet)['geneID_annotation'] must be either 'ENTREZID' or 'SYMBOL'. You will need to annotate yourself the IcaSet object if you want to use different IDs.

Value

The modified argument icaSet, with filled attributes SByGene and datByGene.

Author(s)

Anne Biton

See Also

annotFeaturesComp

Examples

```
#load data
data(icaSetCarbayo)
require(hgu133a.db)

# run annotation of the features into gene Symbols as specified in 'typeID(icaSetCarbayo)["geneID_annotation"
# using package hgu133a.db as defined in 'annotation(icaSetMainz)'
icaSetCarbayo <- annotInGene(icaSet=icaSetCarbayo, params=buildMineICAParams())

## Not run:
#load data
library(breastCancerMAINZ)
data(mainz)
#run ICA
resJade <- runICA(X=exprs(mainz), nbComp=5, method = "JADE", maxit=10000)
```

```
#build params
params <- buildMineICAParams(resPath="mainz/")

#build a new IcaSet object, omitting annotation of the features (runAnnot=FALSE)
#but specifying the element "geneID_annotation" of argument 'typeID'
icaSetMainz <- buildIcaSet(params=params, A=data.frame(resJade$A), S=data.frame(resJade$S),
                           dat=exprs(mainz), pData=pData(mainz),
                    annotation="hgu133a.db", typeID= c(geneID_annotation = "SYMBOL",
                   geneID_biomart = "hgnc_symbol", featureID_biomart = "affy_hg_u133a"),
                           chipManu = "affymetrix", runAnnot=FALSE,
                   mart=useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl"))

#Attributes SByGene is empty and attribute datByGene refers to assayData
SByGene(icaSetMainz)
head(datByGene(icaSetMainz))

# run annotation of the features into gene Symbols as specified in 'typeID(icaSetMainz)["geneID_annotation"]'
# using package hgu133a.db as defined in 'annotation(icaSetMainz)'
icaSetMainz <- annotInGene(icaSet=icaSetMainz, params=params)

## End(Not run)
```

---

annotReciprocal *annotReciprocal*

Description

This function notes edges of a graph as reciprocal or not.

Usage

```
annotReciprocal(dataGraph, file,
  keepOnlyReciprocal = FALSE)
```

Arguments

dataGraph	data.frame which contains the graph description, must have two columns n1 and n2 filled with node IDs, each row denoting there is an edge from n1 to n2.
file	file where the graph description is written
keepOnlyReciprocal	if TRUE dataGraph is restricted to reciprocal edges, else all edges are kept (default).

Value

This function returns the argument dataGraph with an additional column named 'reciprocal' which contains TRUE if the edge described by the row is reciprocal, and FALSE if it is not reciprocal.

Author(s)

Anne Biton

Examples

```
dg <- data.frame(n1=c("A","B","B","C","C","D","E","F"),n2=c("B","G","A","B","D","C","F","E"))
annotReciprocal(dataGraph=dg)
```

---

buildIcaSet *This function builds an object of class* IcaSet.

Description

This function builds an object of class IcaSet.

Usage

```
buildIcaSet(params, A, S, dat, pData = new("data.frame"),
  fData = new("data.frame"), witGenes = new("character"),
  compNames = new("character"),
  refSamples = new("character"),
  annotation = new("character"),
  chipManu = new("character"),
  chipVersion = new("character"), alreadyAnnot = FALSE,
 typeID = c(geneID_annotation = "SYMBOL", geneID_biomart = "hgnc_symbol", featureID_biomart = "'
  runAnnot = TRUE, organism = "Human",
  mart = new("Mart"))
```

Arguments

params	An object of class MineICAParams containing the parameters of the analysis
A	The mixing matrix of the ICA decomposition (of dimension samples x components).
S	The source matrix of the ICA decomposition (of dimension features x components).
dat	The data matrix the ICA was applied to (of dimension features x samples).
pData	Phenotype data, a data.frame which contains the sample informations of dimension samples x annotations.
fData	Feature data, a data.frame which contrains the feature descriptions of dimensions features x annotations.
witGenes	A vector of witness genes. They are representative of the expression behavior of the contributing genes of each component. If missing or NULL, they will be automatically attributed using function selectWitnessGenes.
compNames	A vector of component labels.
refSamples	A vector of reference sample IDs (e.g the "normal" samples).
annotation	An annotation package (e.g a ".db" package specific to the microarray used to generate dat)
chipManu	If microarray data, the manufacturer: either 'affymetrix' or 'illumina'.
chipVersion	For illumina microarrays: the version of the microarray.
alreadyAnnot	TRUE if the feature IDs contained in the row names of dat and S already correspond to the final level of annotation (e.g if they are already gene IDs). In that case, no annotation is performed.

typeID — A character vector specifying the annotation IDs, it includes three elements :

geneID_annotation the IDs from the package to be used to annotate the features into genes. It will be used to fill the attributes datByGene and SByGene of the icaSet. It must match one of the objects the corresponding package supports (you can access the list of objects by typing ls("package:packagename")). If no annotation package is provided, this element is not useful.

geneID_biomart the type of gene IDs, as available in listFilters(mart); where mart is specified as described in useMart. If you have directly built the IcaSet at the gene level (i.e if no annotation package is used), featureID_biomart and geneID_biomart will be identical.

featureID_biomart the type of feature IDs, as available in listFilters(mart); where mart is specified as described in function useMart. Not useful if you work at the gene level.

runAnnot — If TRUE, icaSet is annotated with function annotInGene.

organism — The organism the data correspond to.

mart — The mart object (database and dataset) used for annotation, see function useMart of package biomaRt

Value

An object of class IcaSet

Author(s)

Anne Biton

See Also

selectWitnessGenes, annotInGene

Examples

```
dat <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat) <- paste("g", 1:1000, sep="")
colnames(dat) <- paste("s", 1:10, sep="")

## build a data.frame containing sample annotations
annot <- data.frame(type=c(rep("a",5),rep("b",5)))
rownames(annot) <- colnames(dat)

## run ICA
resJade <- runICA(X=dat, nbComp=3, method = "JADE")

## build params
params <- buildMineICAParams(resPath="toy/")

## build IcaSet object
icaSettoy <- buildIcaSet(params=params, A=data.frame(resJade$A), S=data.frame(resJade$S),
                         dat=dat, pData=annot, alreadyAnnot=TRUE)
params <- icaSettoy$params
icaSettoy <- icaSettoy$icaSet

## Not run:
## load data
```

```
library(breastCancerMAINZ)
data(mainz)

## run ICA
resJade <- runICA(X=dataMainz, nbComp=10, method = "JADE", maxit=10000)

## build params
params <- buildMineICAParams(resPath="mainz/")

## build IcaSet object

# fill typeID, Mainz data originate from affymetrix HG-U133a  microarray and are indexed by probe sets
# we want to annotate the probe sets into Gene Symbols
typeIDmainz <-  c(geneID_annotation="SYMBOL", geneID_biomart="hgnc_symbol", featureID_biomart="affy_hg_u133a

icaSetMainz <- buildIcaSet(params=params, A=data.frame(resJade$A), S=data.frame(resJade$S),
                           dat=exprs(mainz), pData=pData(mainz),
                         annotation="hgu133a.db", typeID= c(geneID_annotation = "SYMBOL",
                        geneID_biomart = "hgnc_symbol", featureID_biomart = "affy_hg_u133a"),
                           chipManu = "affymetrix", runAnnot=TRUE,
                        mart=useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl"))

## End(Not run)
```

`buildMineICAParams` *Creates an object of class MineICAParams*

Description

This function builds an object of class `MineICAParams`. It contains the parameters that will be used by function `runAn` to analyze the ICA decomposition contained in an object of class `IcaSet`.

Usage

```
buildMineICAParams(Sfile = new("character"),
  Afile = new("character"), datfile = new("character"),
  annotfile = new("character"), resPath = "", genesPath,
  annot2col = new("character"), pvalCutoff = 0.05,
  selCutoff = 3)
```

Arguments

`Sfile`	A txt file containing the Source matrix S.
`Afile`	A txt file containing the Mixing matrix A.
`datfile`	A txt file containing the data (e.g expression data) on which the decomposition was calculated.
`annotfile`	Either a "rda" or "txt" file containing the annotation data for the samples (must be of dimensions samples x annotations).
`resPath`	The path where the outputs of the analysis will be written, default is the current directory.
`genesPath`	The path _within_ the resPath where the gene projections will be written. If missing, will be automatically attributed as `resPath`/ProjByComp/.

annot2col	A vector of colors indexed by annotation levels. If missing, will be automatically attributed using function annot2Color.
pvalCutoff	The cutoff used to consider a p-value significant, default is 0.05.
selCutoff	The cutoff applied to the absolute feature/gene projection values to consider them as contributors, default is 3. Must be either of length 1 and the same treshold is applied to all components, or of length equal to the number of components in order to a specific threshold is for each component.

Value

An object of class MineICAParams

Author(s)

Anne Biton

See Also

MineICAParams, runAn

Examples

```
## define default parameters and fill resPath
params <- buildMineICAParams(resPath="resMineICACarbayo/")

## change the default cutoff for selection of contribugint genes/features
params <- buildMineICAParams(resPath="resMineICACarbayo/", selCutoff=4)
```

---

clusterFastICARuns *Run of fastICA and JADE algorithms*

Description

This function runs the fastICA algorithm several times with random initializations. The obtained components are clustered and the medoids of these clusters are used as the final estimates. The returned estimates are ordered by decreasing Iq values which measure the compactness of the clusters (see details).

Usage

```
clusterFastICARuns(X, nbComp, nbIt = 100,
  alg.type = c("deflation", "parallel"),
  fun = c("logcosh", "exp"), maxit = 500, tol = 10^-6,
  funClus = c("hclust", "agnes", "pam", "kmeans"),
  row.norm = FALSE, bootstrap = FALSE, ...)
```

Arguments

X	A data matrix with n rows representing observations (e.g genes) and p columns representing variables (e.g samples).
nbComp	The number of components to be extracted.
nbIt	The number of iterations of FastICA
alg.type	If alg.type="parallel" the components are extracted simultaneously (the default), if alg.type="deflation" the components are extracted one at a time, see fastICA.
fun	The functional form of the G function used in the approximation to neg-entropy (see 'details' of the help of function fastICA).
row.norm	a logical value indicating whether rows of the data matrix X should be standardized beforehand (see help of function fastICA)
maxit	The maximum number of iterations to perform.
tol	A positive scalar giving the tolerance at which the un-mixing matrix is considered to have converged.
funClus	The clustering function to be used to cluster the estimates
bootstrap	if TRUE the data is bootstraped before each fastICA iteration, else (default) only random initializations are done
...	Additional parameters for codefunClus

Details

This function implements in R fastICA iterations followed by a clustering step, as defined in the matlab package 'icasso'. Among the indices computed by icasso, only the Iq index is currently computed. As defined in 'icasso', the Iq index measures the difference between the intra-cluster similarity and the extra-cluster similiarity. No visualization of the clusters is yet available.

If bootstrap=TRUE a bootstrap (applied to the observations) is used to perturb the data before each iteration, then function fastICA is applied with random initializations.

By default, in 'icasso', agglomerative hierarchical clustering with average linkage is performed. To use the same clustering, please use funClus="hclust" and method="average". But this function also allows you to apply the clustering of your choice among kmeans, pam, hclust, agnes by specifying funClus and adding the adequat additional parameters.

See details of the functions fastICA.

Value

A list consisting of:

A the estimated mixing matrix

S the estimated source matrix, itemWthe estimated unmixing matrix,

Iq Iq indices.

Author(s)

Anne Biton

Examples

```
## generate a data
set.seed(2004);
M <- matrix(rnorm(5000*6,sd=0.3),ncol=10)
M[1:100,1:3] <- M[1:100,1:3] + 2
M[1:200,1:3] <- M[1:200,4:6] +1

## Random initializations are used for each iteration of FastICA
## Estimates are clustered using hierarchical clustering with average linkage
res <- clusterFastICARuns(X=M, nbComp=2, alg.type="deflation",
                          nbIt=3, funClus="hclust", method="average")

## Data are boostraped before each iteration and random initializations
## are used for each iteration of FastICA
## Estimates are clustered using hierarchical clustering with ward
res <- clusterFastICARuns(X=M, nbComp=2, alg.type="deflation",
                          nbIt=3, funClus="hclust", method="ward")
```

---

clusterSamplesByComp *Cluster samples from an IcaSet*

Description

This function allows to cluster samples according to the results of an ICA decomposition. One clustering is run independently for each component.

Usage

```
clusterSamplesByComp(icaSet, params,
  funClus = c("Mclust", "kmeans", "pam", "pamk", "hclust", "agnes"),
  filename, clusterOn = c("A", "S"),
  level = c("genes", "features"), nbClus,
  metric = "euclidean", method = "ward", ...)
```

Arguments

icaSet	An IcaSet object
params	A MineICAParams object
funClus	The function to be used for clustering, must be one of c("Mclust","kmeans","pam","pamk","hclu
filename	A file name to write the results of the clustering in
clusterOn	Specifies the matrix used to apply clustering: "A": the clustering is performed in one dimension, on the vector of sample contributions, "S": the clustering is performed on the original data restricted to the contributing individuals.
level	The level of projections to be used when clusterOn="S", either "features" or "genes".
nbClus	The number of clusters to be computed, either a single number or a numeric vector whose length equals the number of components. If missing (only allowed if funClus is one of c("Mclust","pamk"))

metric	Metric used in pam and hclust, default is "euclidean"
method	Method of hierarchical clustering, used in hclust and agnes
...	Additional parameters required by the clustering function funClus.res <- clusterSamplesByComp(icaSet=icaSetCarbayo, params=params, funClus="kmeans",

Value

A list consisting of three elements

clus: a list specifying the sample clustering for each component,

resClus: the complete output of the clustering function,

funClus: the function used to perform the clustering.

. When clusterOn="S", if some components were not used because no contributing elements is selected using the cutoff, the icaSet with the corresponding component deleted is also returned.

Author(s)

Anne Biton

See Also

Mclust, kmeans, pam, pamk, hclust, agnes, cutree

Examples

```
data(icaSetCarbayo)
params <- buildMineICAParams(resPath="carbayo/", selCutoff=4)

## cluster samples according to their contributions
# using Mclust without a number of clusters
res <- clusterSamplesByComp(icaSet=icaSetCarbayo, params=params, funClus="Mclust",
                            clusterOn="A", filename="clusA")

# using kmeans
res <- clusterSamplesByComp(icaSet=icaSetCarbayo, params=params, funClus="kmeans",
                            clusterOn="A", nbClus=2, filename="clusA")
```

---

clusterSamplesByComp_multiple

Cluster samples from an IcaSet

Description

This function allows to cluster samples according to the results of an ICA decomposition. Several clustering functions and several levels of data for clustering can be performed by the function.

Usage

```
clusterSamplesByComp_multiple(icaSet, params,
  funClus = c("Mclust", "kmeans", "pam", "pamk", "hclust", "agnes"),
  filename, clusterOn = c("A", "S"),
  level = c("genes", "features"), nbClus,
  metric = "euclidean", method = "ward", ...)
```

Arguments

icaSet	An IcaSet object
params	A MineICAParams object
funClus	The function to be used for clustering, must be several of c("Mclust","kmeans","pam","pamk","hc
filename	A file name to write the results of the clustering in
clusterOn	Specifies the matrix used to apply clustering, can be several of: "A": the clustering is performed in one dimension, on the vector of sample contributions, "S": the clustering is performed on the original data restricted to the contributing individuals.
level	The level of projections to be used when clusterOn="S", either "features" or "genes".
nbClus	The number of clusters to be computed, either a single number or a numeric vector whose length equals the number of components. If missing (only allowed if funClus is one of c("Mclust","pamk"))
metric	Metric used in pam and hclust, default is "euclidean"
method	Method of hierarchical clustering, used in hclust and agnes
...	Additional parameters required by the clustering function funClus.

Details

One clustering is run independently for each component.

Value

A list consisting of three elements

clus: a data.frame specifying the sample clustering for each component using the different ways of clustering,

resClus: the complete output of the clustering function(s),

comparClus: the adjusted Rand indices, used to compare the clusterings obtained for a same component.

Author(s)

Anne

See Also

Mclust, adjustedRandIndex, kmeans, pam, pamk, hclust, agnes, cutree

Examples

```
data(icaSetCarbayo)
params <- buildMineICAParams(resPath="carbayo/", selCutoff=3)

## compare kmeans clustering applied to A and data restricted to the contributing genes
## on components 1 to 3
res <- clusterSamplesByComp_multiple(icaSet=icaSetCarbayo[,,1:3], params=params, funClus="kmeans",
                                     nbClus=2, clusterOn=c("A","S"), level="features")
head(res$clus)
```

clusVarAnalysis *Tests association between clusters of samples and variables*

Description

From a clustering of samples performed according to their contribution to each component, this function computes the chi-squared test of association between each variable level and the cluster, and summarizes the results in an HTML file.

Usage

```
clusVarAnalysis(icaSet, params, resClus, keepVar,
  keepComp, funClus = "",
  adjustBy = c("none", "component", "variable"),
  method = "BH", doPlot = FALSE,
  cutoff = params["pvalCutoff"],
  path = paste(resPath(params), "clus2var/", sep = ""),
  onlySign = TRUE, typeImage = "png",
  testBy = c("variable", "level"), filename)
```

Arguments

icaSet	An object of class IcaSet
params	An object of class MineICAParams providing the parameters of the analysis
resClus	A list of numeric vectors indexed by sample IDs, which specifies the sample clusters. There must be one clustering by component of icaSet. The names of the list must correspond to the component indices.
keepVar	The variable labels to be considered, i.e a subset of the variables of icaSet available in varLabels(icaSet).
keepComp	A subset of components available in indComp(icaSet) to be considered, if missing all components are used.
funClus	The name of the function used to perform the clustering (just for text in written files).
adjustBy	The way the p-values of the Wilcoxon and Kruskal-Wallis tests should be corrected for multiple testing: "none" if no p-value correction has to be done, "component" if the p-values have to be corrected by component, "variable" if the p-values have to be corrected by variable.
testBy	Chi-square tests of association can be performed either by "variable" (one test by variable, default) or by variable "level" (as many tests as there are annotation levels).
method	The correction method, see p.adjust for details, default if "BH" for Benjamini & Hochberg.
doPlot	If TRUE, the barplots showing the distribution of the annotation levels among the clusters are plotted and the results are provided in an HTML file 'cluster2annot.htm', else no plot is created.
cutoff	The threshold for statistical significance.
filename	File name for test results, if doPlot=TRUE will be an HTML file else will be a 'txt' file. If missing when doPlot=TRUE, will be "clusVar".

path	A directory _within resPath(params)_ where the outputs are saved if doPlot=TRUE, default is 'cluster2annot/'.
onlySign	If TRUE (default), only the significant results are plotted.
typeImage	The type of image file where each plot is saved.

Details

When doPlot=TRUE, this function writes an HTML file containing the results of the tests as a table of dimension 'variable levels x components' which contains the p-values of the tests. When a p-value is considered as significant according to the threshold cutoff, it is written in bold and filled with a link pointing to the corresponding barplot displaying the distribution of the clusters across the levels of the variables.

One image is created by plot and located into the sub-directory "plots/" of path. Each image is named by index-of-component_var.png

Value

This function returns a list whose each element gives, for each component, the results of the association chi-squared tests between the clusters and the annotation levels.

Author(s)

Anne Biton

See Also

clusterSamplesByComp

Examples

```
## load an example of IcaSet
data(icaSetCarbayo)
## build object of class MineICAParams
params <- buildMineICAParams(resPath="carbayo/")

## cluster samples according to the columns of the mixing matrix A with kmeans in 2 groups
resClus <- clusterSamplesByComp(icaSet=icaSetCarbayo, params=params, funClus="kmeans",
                                clusterOn="A", nbClus=2)$clus

## specify directory for the function outputs (here same directory as the default one)
## this directory will be created by the function in resPath(params)
dir <- "clus2var/"

## compute chi-square tests of association, p-value are not adjusted (adjustBy="none"),
# test results are written in txt format (doPlot=FALSE and filename not missing)
resChi <- clusVarAnalysis(icaSet=icaSetCarbayo, params=params, resClus=resClus, funClus="kmeans",
                        adjustBy="none", doPlot=FALSE, path=dir, filename="clusVarTests")

## Not run:
## compute chi-square tests of association, p-value are not adjusted (adjustBy="none"),
# write results and plots in HTML files (doPlot=TRUE)
resChi <- clusVarAnalysis(icaSet=icaSetCarbayo, params=params, resClus=resClus, funClus="kmeans",
                         path=dir, adjustBy="none", doPlot=TRUE, filename="clusVarTests")

## compute chi-square tests of association by only considering a subset of components and variables,
```

```
# adjust p-values by component  (adjustBy="component"),
# do not write results (doPlot=FALSE and filename is missing).
resChi <- clusVarAnalysis(icaSet=icaSetCarbayo, params=params, resClus=resClus, keepComp = 1:10,
                          keepVar=c("GENDER","STAGE"), funClus="kmeans", adjustBy="none",
                           doPlot=FALSE)

## End(Not run)
```

`compareAn` *Comparison of IcaSet objects using correlation*

Description

Compare `IcaSet` objects by computing the correlation between either projection values of common features or genes, or contributions of common samples.

Usage

```
compareAn(icaSets, labAn,
  type.corr = c("pearson", "spearman"), cutoff_zval = 0,
  level = c("samples", "features", "genes"))
```

Arguments

`icaSets`	list of IcaSet objects, e.g results of ICA decompositions obtained on several datasets.
`labAn`	vector of names for each icaSet, e.g the the names of the datasets on which were calculated the decompositions.
`type.corr`	Type of correlation to compute, either `'pearson'` or `'spearman'`.
`cutoff_zval`	either NULL or 0 (default) if all genes are used to compute the correlation between the components, or a threshold to compute the correlation on the genes that have at least a scaled projection higher than cutoff_zval. Will be used only when correlations are calculated on S or SByGene.
`level`	Data level of the `IcaSet` objects on which is applied the correlation. It must correspond to a feature shared by the IcaSet objects: `'samples'` if they were applied to common samples (correlations are computed between matrix `A`), `'features'` if they were applied to common features (correlations are computed between matrix `S`), `'genes'` if they share gene IDs after annotation into genes (correlations are computed between matrix `SByGene`).

Details

The user must carefully choose the object on which the correlation will be computed. If `level='samples'`, the correlations are based on the mixing matrices of the ICA decompositions (of dimension samples x components). `'A'` will be typically chosen when the ICA decompositions were computed on the same dataset, or on datasets that include the same samples. If `level='features'` is chosen, the correlation is calculated between the source matrices (of dimension features x components) of the ICA decompositions. `'S'` will be typically used when the ICA decompositions share common features (e.g same microarrays). If `level='genes'`, the correlations are calculated on the attributes

'SByGene' which store the projections of the annotated features. 'SByGene' will be typically chosen when ICA were computed on datasets from different technologies, for which comparison is possible only after annotation into a common ID, like genes.

cutoff_zval is only used when level is one of c('genes','features'), in order to restrict the correlation to the contributing features or genes.

When cutoff_zval is specified, for each pair of components, genes or features that are included in the circle of center 0 and radius cutoff_zval are excluded from the computation of the correlation.

It must be taken into account by the user that if cutoff_zval is different from NULL or 0, the computation will be much slowler since each pair of component is treated individually.

Value

A list whose length equals the number of pairs of IcaSet and whose elements are outputs of function cor2An.

Author(s)

Anne Biton

See Also

cor2An

Examples

```
dat1 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat1) <- paste("g", 1:1000, sep="")
colnames(dat1) <- paste("s", 1:10, sep="")
dat2 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat2) <- paste("g", 1:1000, sep="")
colnames(dat2) <- paste("s", 1:10, sep="")

## run ICA
resJade1 <- runICA(X=dat1, nbComp=3, method = "JADE")
resJade2 <- runICA(X=dat2, nbComp=3, method = "JADE")

## build params
params <- buildMineICAParams(resPath="toy/")

## build IcaSet object
icaSettoy1 <- buildIcaSet(params=params, A=data.frame(resJade1$A), S=data.frame(resJade1$S),
                          dat=dat1, alreadyAnnot=TRUE)$icaSet
icaSettoy2 <- buildIcaSet(params=params, A=data.frame(resJade2$A), S=data.frame(resJade2$S),
                          dat=dat2, alreadyAnnot=TRUE)$icaSet

listPairCor <- compareAn(icaSets=list(icaSettoy1,icaSettoy2), labAn=c("toy1","toy2"),
                         type.corr="pearson", level="genes", cutoff_zval=0)


## Not run:
#### Comparison of 2 ICA decompositions obtained on 2 different gene expression datasets.
## load the two datasets
library(breastCancerMAINZ)
library(breastCancerVDX)
data(mainz)
```

```
data(vdx)

## Define a function used to build two examples of IcaSet objects
treat <- function(es, annot="hgu133a.db") {
   es <- selectFeatures_IQR(es,10000)
   exprs(es) <- t(apply(exprs(es),1,scale,scale=FALSE))
   colnames(exprs(es)) <- sampleNames(es)
   resJade <- runICA(X=exprs(es), nbComp=10, method = "JADE", maxit=10000)
  resBuild <- buildIcaSet(params=buildMineICAParams(), A=data.frame(resJade$A), S=data.frame(resJade$S),
                       dat=exprs(es), pData=pData(es), refSamples=character(0),
                       annotation=annot, typeID= typeIDmainz,
                       chipManu = "affymetrix", mart=mart)
   icaSet <- resBuild$icaSet
}
## Build the two IcaSet objects
icaSetMainz <- treat(mainz)
icaSetVdx <- treat(vdx)

## The pearson correlation is used as a measure of association between the gene projections
# on the different components (type.corr="pearson").
listPairCor <- compareAn(icaSets=list(icaSetMainz,icaSetVdx),
labAn=c("Mainz","Vdx"), type.corr="pearson", level="genes", cutoff_zval=0)

## Same thing but adding a selection of genes on which the correlation between two components is computed:
# when considering pairs of components, only projections whose scaled values are not located within
# the circle of radius 1 are used to compute the correlation (cutoff_zval=1).
listPairCor <-  compareAn(icaSets=list(icaSetMainz,icaSetVdx),
labAn=c("Mainz","Vdx"), type.corr="pearson", cutoff_zval=1, level="genes")

## End(Not run)
```

`compareAn2graphfile` *compareAn2graphfile*

Description

This function builds a correlation graph from the outputs of function `compareAn`.

Usage

```
compareAn2graphfile(listPairCor, useMax = TRUE,
  cutoff = NULL, useVal = c("cor", "pval"), file = NULL)
```

Arguments

`listPairCor`	The output of the function `compareAn`, containing the correlation between several pairs of objects of class `IcaSet`.
`useMax`	If TRUE, the graph is restricted to edges that correspond to maximum score, see details
`cutoff`	Cutoff used to select pairs that will be included in the graph.
`useVal`	The value on which is based the graph, either `"cor"` for correlation or `"pval"` for p-values of correlation tests.
`file`	File name.

Details

When correlations are considered (useVal="cor"), absolute values are used since the components have no direction.

If useMax is TRUE each component is linked to the most correlated component of each different IcaSet.

If cutoff is specified, only correlations exceeding this value are taken into account during the graph construction. For example, if cutoff is 1, only relationships between components that correspond to a correlation value larger than 1 will be included.

When useVal="pval" and useMax=TRUE, the minimum value is taken instead of the maximum.

Value

A data.frame with the graph description, has two columns n1 and n2 filled with node IDs, each row denotes that there is an edge from n1 to n2. Additional columns quantify the strength of association: correlation (cor), p-value (pval), (1-abs(cor)) (distcor), log10-pvalue (logpval).

Author(s)

Anne Biton

See Also

compareAn, cor2An

Examples

```
dat1 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat1) <- paste("g", 1:1000, sep="")
colnames(dat1) <- paste("s", 1:10, sep="")
dat2 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat2) <- paste("g", 1:1000, sep="")
colnames(dat2) <- paste("s", 1:10, sep="")

## run ICA
resJade1 <- runICA(X=dat1, nbComp=3, method = "JADE")
resJade2 <- runICA(X=dat2, nbComp=3, method = "JADE")

## build params
params <- buildMineICAParams(resPath="toy/")

## build IcaSet object
icaSettoy1 <- buildIcaSet(params=params, A=data.frame(resJade1$A), S=data.frame(resJade1$S),
                          dat=dat1, alreadyAnnot=TRUE)$icaSet
icaSettoy2 <- buildIcaSet(params=params, A=data.frame(resJade2$A), S=data.frame(resJade2$S),
                          dat=dat2, alreadyAnnot=TRUE)$icaSet

resCompareAn <- compareAn(icaSets=list(icaSettoy1,icaSettoy2), labAn=c("toy1","toy2"),
                         type.corr="pearson", level="genes", cutoff_zval=0)

## Build a graph where edges correspond to maximal correlation value (useVal="cor"),
compareAn2graphfile(listPairCor=resCompareAn, useMax=TRUE, useVal="cor", file="myGraph.txt")


## Not run:
```

```
#### Comparison of 2 ICA decompositions obtained on 2 different gene expression datasets.
## load the two datasets
library(breastCancerMAINZ)
library(breastCancerVDX)
data(mainz)
data(vdx)

## Define a function used to build two examples of IcaSet objects
treat <- function(es, annot="hgu133a.db") {
   es <- selectFeatures_IQR(es,10000)
   exprs(es) <- t(apply(exprs(es),1,scale,scale=FALSE))
   colnames(exprs(es)) <- sampleNames(es)
   resJade <- runICA(X=exprs(es), nbComp=10, method = "JADE", maxit=10000)
  resBuild <- buildIcaSet(params=buildMineICAParams(), A=data.frame(resJade$A), S=data.frame(resJade$S),
                       dat=exprs(es), pData=pData(es), refSamples=character(0),
                       annotation=annot, typeID= typeIDmainz,
                       chipManu = "affymetrix", mart=mart)
   icaSet <- resBuild$icaSet
}
## Build the two IcaSet objects
icaSetMainz <- treat(mainz)
icaSetVdx <- treat(vdx)

## Compute correlation between every pair of IcaSet objects.
resCompareAn <- compareAn(icaSets=list(icaSetMainz,icaSetVdx),
labAn=c("Mainz","Vdx"), type.corr="pearson", level="genes", cutoff_zval=0)

## Same thing but adding a selection of genes on which the correlation between two components is computed:
# when considering pairs of components, only projections whose scaled values are not located within
# the circle of radius 1 are used to compute the correlation (cutoff_zval=1).
resCompareAn <-  compareAn(icaSets=list(icaSetMainz,icaSetVdx),
labAn=c("Mainz","Vdx"), type.corr="pearson", cutoff_zval=1, level="genes")

## Build a graph where edges correspond to maximal correlation value (useVal="cor"),
## i.e, component A of analysis i is linked to component B of analysis j,
## only if component B is the most correlated component to A amongst all component of analysis j.
compareAn2graphfile(listPairCor=resCompareAn, useMax=TRUE, useVal="cor", file="myGraph.txt")

## Restrict the graph to correlation values exceeding 0.4
compareAn2graphfile(listPairCor=resCompareAn, useMax=FALSE, cutoff=0.4,  useVal="cor", file="myGraph.txt")


## End(Not run)
```

---

compareGenes *Union and intersection of contributing genes*

Description

Compute and annotate the intersection or union between contributiong genes of components originating from different IcaSet objects.

Usage

```
compareGenes(keepCompByIcaSet, icaSets, lab, cutoff = 0,
```

```
type = c("union", "intersection"), annotate = TRUE,
file,
mart = useMart("ensembl", "hsapiens_gene_ensembl"))
```

Arguments

icaSets	List of IcaSet objects, the geneNames of the IcaSet objects must be from the same type (e.g, gene Symbols).
keepCompByIcaSet	Indices of the components to be considered in each IcaSet.
lab	The names of the icaSets (e.g the names of the datasets they originate from).
cutoff	The cutoff (on the absolute centered and scaled projections) above which the genes have to be considered.
type	"intersection" to restrict the list of genes to the ones that are common between all datasets, or "union" to consider all the union of genes available across the datasets.
annotate	If TRUE (default) the genes are annotated using function writeGenes.
file	The HTML file name where the genes and their annotations are written, default is typeGenes_lab1-i_lab2-j_... where i and j are the component indices contained in keepCompByIcaSet.
mart	The mart object (database and dataset) used for annotation, see function useMart of package biomaRt.

Value

A data.frame containing

typeID(icaSets[[1]])['geneID_biomart']: the gene IDs,

median_rank the median of the ranks of each gene across the IcaSet objects,

analyses the labels of the IcaSet objects in which each gene is above the given cutoff

min_rank the minimum of the ranks of each gene across the IcaSet objects,

ranks the ranks of each gene in each IcaSet where it is available,

scaled_proj the centered and reduced projection of each gene in each IcaSet where it is available.

Author(s)

Anne Biton

See Also

writeGenes

Examples

```
## Not run:
data(icaSetCarbayo)
mart <- useMart("ensembl", "hsapiens_gene_ensembl")

## comparison of two components
## here the components come from the same IcaSet for convenience
## but they must come from different IcaSet in practice.
```

```
compareGenes(keepCompByIcaSet = c(9,4), icaSets = list(icaSetCarbayo, icaSetCarbayo),
             lab=c("Carbayo", "Carbayo2"), cutoff=3, type="union",  mart=mart)


## End(Not run)
```

---

cor2An *Correlation between two matrices*

Description

This function measures the correlation between two matrices containing the results of two decompositions.

Usage

```
cor2An(mat1, mat2, lab,
  type.corr = c("pearson", "spearman"), cutoff_zval = 0)
```

Arguments

mat1	matrix of dimension features/genes x number of components, e.g the results of an ICA decomposition
mat2	matrix of dimension features/genes x number of components, e.g the results of an ICA decomposition
lab	The vector of labels for mat1 and mat2, e.g the the names of the two datasets on which were calculated the two decompositions
type.corr	Type of correlation, either 'pearson' or 'spearman'
cutoff_zval	cutoff_zval: 0 (default) if all genes are used to compute the correlation between the components, or a threshold to compute the correlation on the genes that have at least a scaled projection higher than cutoff_zval.

Details

Before computing the correlations, the components are scaled and restricted to common row names.

It must be taken into account by the user that if cutoff_zval is different from NULL or zero, the computation will be slowler since each pair of component is treated individually.

When cutoff_zval is specified, for each pair of components, genes that are included in the circle of center 0 and radius cutoff_zval are excluded from the computation of the correlation between the gene projection of the two components.

Value

This function returns a list consisting of:

cor	a matrix of dimensions '(nbcomp1+nbcomp2) x (nbcomp1*nbcomp2)', containing the correlation values between each pair of components,
pval	a matrix of dimension '(nbcomp1+nbcomp2) x (nbcomp1*nbcomp2)', containing the p-value of the correlation tests for each pair of components,
inter	the intersection between the features/genes of mat1 and mat2,
labAn	the labels of the compared matrices.

Author(s)

Anne Biton

See Also

rcorr, cor.test, compareAn

Examples

```
cor2An(mat1=matrix(rnorm(10000),nrow=1000,ncol=10), mat2=matrix(rnorm(10000),nrow=1000,ncol=10),
       lab=c("An1","An2"), type.corr="pearson")
```

---

correl2Comp *correl2Comp*

Description

This function computes the correlation between two components.

Usage

```
correl2Comp(comp1, comp2, type.corr = "pearson", plot = FALSE,
    cutoff_zval = 0, test = FALSE, alreadyTreat = FALSE)
```

Arguments

comp1	The first component, a vector of projections or contributions indexed by labels
comp2	The second component, a vector of projections or contributions indexed by labels
type.corr	Type of correlation to be computed, either 'pearson' or 'spearman'
plot	if TRUE, plot comp1 vs comp2
cutoff_zval	either NULL or 0 (default) if all genes are used to compute the correlation between the components, or a threshold to compute the correlation on the genes that have at least a scaled projection higher than cutoff_zval.
test	if TRUE the correlation test p-value is returned instead of the correlation value
alreadyTreat	if TRUE comp1 and comp2 are considered as being already treated (i.e scaled and restricted to common elements)

Details

Before computing the correlation, the components are scaled and restricted to common labels. When cutoff_zval is different from 0, the elements that are included in the circle of center 0 and radius cutoff_zval are not taken into account during the computation of the correlation.

Value

This function returns either the correlation value or the p-value of the correlation test.

Author(s)

Anne Biton

dat *Retrieve and set data from IcaSet*

Description

These generic functions access and set the attributes dat stored in an object of class `IcaSet`.

Usage

```
dat(object)
dat(object) <- value
datByGene(object)
datByGene(object) <- value
geneNames(object)
```

Arguments

`object`	object of class `IcaSet`
`value`	Matrix with rows representing features or genes and columns samples.

Value

`dat` and `datByGene` return a matrix containing measured values (e.g expression data) indexed by features and genes, respectively. `geneNames` returns the names of the genes, i.e the row names of `datByGene`.

Author(s)

Anne

dataCarbayo *Carbayo expression data*

Description

Contains bladder cancer expression data based on on HG-U133A Affymetrix microarrays. The data include 93 samples, were normalized with MAS5 by the authors of the paper using Quantile normalization and log2-transformation. They are restricted to the 10000 most variable probe sets.

Author(s)

Anne Biton

References

http://jco.ascopubs.org/content/24/5/778/suppl/DC1

getComp — *Retrieve feature and sample values on a component stored in an* IcaSet *object.*

Description

This generic function retrieves, from an IcaSet object, the feature projections (contained in attribute S) and sample contributions (contained in attribute A) corresponding to a specific component.

Usage

```
getComp(object, level, ind)
```

Arguments

object	Object of class IcaSet.
level	Either "features" to retrieve projections contained in S, or "genes" to retrieve projections contained in SByGene.
ind	The index of the component to be retrieved.

Value

getComp returns a list containing two elements:

proj: the feature or gene projections on the given component,

contrib: the sample contributions on the given component.

Author(s)

Anne Biton

See Also

IcaSet-class

getProj — *Extract projection values*

Description

Extract projection values of a given set of IDs on a subset of components.

Usage

```
getProj(icaSet, ids, keepComp,
  level = c("features", "genes"))
```

Arguments

icaSet	An object of class IcaSet
ids	feature or gene IDs
keepComp	Index of the components to be conserved, must be in indComp(icaSet)
level	The level of projections to be extracted, either "features" or "genes"

Value

A vector or a list of projection values

Author(s)

Anne Biton

Examples

```
## load an example of IcaSet
data(icaSetCarbayo)

##get the projection of your favorite proliferation genes
#on all components
getProj(icaSetCarbayo, ids=c("TOP2A","CDK1","CDC20"), level="genes")

#on some components
getProj(icaSetCarbayo, ids=c("TOP2A","CDK1","CDC20"),
keepComp=c(1,6,9,12),level="genes")

##get the gene projection values on the sixth component
getProj(icaSetCarbayo, keepComp=6,level="genes")
```

---

hgOver *Output of* hyperGtest

Description

Example of output of function hyperGtest.

Author(s)

Anne Biton

hypergeoAn *Runs an enrichment analysis per component using package GOstats.*

Description

Runs an enrichment analysis of the contributing genes associated with each component, using the function hyperGTest of package GOstats. The easiest way to run enrichment analysis is to use function runEnrich.

Usage

```
hypergeoAn(icaSet, params,
  path = paste(resPath(params), "GOstatsEnrichAnalysis/", sep = "/"),
  SlistSel, hgCutoff = 0.01, db = "go", onto = "BP",
  cond = TRUE, universe, entrez2symbol)
```

Arguments

icaSet	An object of class IcaSet
params	An object of class MineICAParams containing the parameters of the analysis
path	The path where results will be saved
SlistSel	A list of contributing gene projection values per component. Each element of the list corresponds to a component and is restricted to the features or genes exceeding a given threshold. If missing, is computed by the function.
hgCutoff	The p-value threshold
db	The database to be used ("GO" or "KEGG")
onto	A character specifying the GO ontology to use. Must be one of "BP", "CC", or "MF", see GOHyperGParams. Only used when argument db is "GO".
cond	A logical indicating whether the calculation should conditioned on the GO structure, see GOHyperGParams.
universe	The universe for the hypergeometric tests, see GOHyperGParams.
entrez2symbol	A vector of all gene Symbols involved in the analysis indexed by their Entrez Gene IDs. It is only used when annotation(params) is empty, and allows to associate gene sets to Symbols.

Details

An annotation package must be available in annotation(icaSet) to provide the contents of the gene sets. If none corresponds to the technology you deal with, please choose the org.*.eg.db package according to the organism (for example org.Hs.eg.db for Homo sapiens). Save results of the enrichment tests in a '.rda' file located in path/db/onto/zvalCutoff(params).

Author(s)

Anne Biton

See Also

runEnrich, xtable, useMart, hyperGTest, GOHyperGParams, mergeGostatsResults

Examples

```
## Not run:
## load an example of IcaSet
data(icaSetCarbayo)

## define params
# Use threshold 3 to select contributing genes.
# Results of enrichment analysis will be written in path 'resPath(params)/GOstatsEnrichAnalysis'
params <- buildMineICAParams(resPath="~/resMineICACarbayo/", selCutoff=3)

## Annotation package for IcaSetCarbayo is hgu133a.db.
# check annotation package
annotation(icaSetCarbayo)

## Define universe, i.e the set of EntrezGene IDs mapping to the feature IDs of the IcaSet object.
universe <- as.character(na.omit(unique(unlist(AnnotationDbi::mget(featureNames(icaSetCarbayo),
                        hgu133aENTREZID, ifnotfound = NA)))))

## Apply enrichement analysis (of the contributing genes) to the first components using gene sets from KEGG.
# Since an annotation package is available, we don't need to fill arg 'entrez2symbol'.
# run the actual enrichment analysis
hypergeoAn(icaSet=icaSetCarbayo[,,1], params=params, db="GO",onto="BP", universe=universe)

## End(Not run)
```

IcaSet *Class to Contain and Describe an ICA decomposition of High-Throughput Data.*

Description

Container for high-throughput data and results of ICA decomposition obtained on these data. `IcaSet` class is derived from `eSet`, and requires a matrix named `dat` as `assayData` member.

Extends

Directly extends class `eSet`.

Creating Objects

```
new("IcaSet")
new("IcaSet",    annotation = character(0),    experimentData = new("MIAME"),    featureData = new("
```

This creates an `IcaSet` with `assayData` implicitly created to contain `dat`.

```
new("IcaSet",    annotation = character(0),    assayData = assayDataNew(dat=new("matrix")),    expe
```

This creates an `IcaSet` with `assayData` provided explicitly.

`IcaSet` instances are usually created through `new("IcaSet", ...)`. Usually the arguments to `new` include `dat` ('features x samples', e.g a matrix of expression data), `phenoData` ('samples x annotations', a matrix of sample annotations), `S` the Source matrix of the ICA decomposition ('features

x comp'), A the Mixing matrix of the ICA decomposition ('samples x comp'), annotation the annotation package, typeID the description of the feature and gene IDs.

The other attributes can be missing, in which case they are assigned default values.

The function buildIcaSet is a more convenient way to create IcaSet instances, and allows to automatically annotate the features.

Slots

Inherited from eSet:

annotation: See eSet

assayData: Contains matrices with equal dimensions, and with column number equal to nrow(phenoData). assayData must contain a matrix dat with rows representing features (e.g., reporters) and columns representing samples. Class:AssayData-class

experimentData: See eSet

featureData: See eSet

phenoData: See eSet

protocolData: See eSet

Specific slot:

organism: Contains the name of the species. Currently only Human ("Human" or "Homo sapiens") and Mouse ("Mouse" or "Mus musculus") are supported. Only used when chipManu="illumina"

mart: An output of useMart of package biomaRt. Only useful if no annotation package is available for argument icaSet.

datByGene: Data.frame containing the data dat where features have been replaced by their annotations (e.g, gene IDs). Rows represent annotations of the features (e.g., gene IDs) and columns represent samples.

A: The mixing matrix of the ICA decomposition, contained in a data.frame whose column number equals the number of components and row number equals nrow(phenoData) (dimension: 'samples x comp').

S: The source matrix of the ICA decomposition, contained in a data.frame whose column number equals the number of components and row number equals nrow(assayData) (dimension: 'features x comp').

SByGene: The matrix Source of the ICA decomposition, contained in a data.frame whose column number equals the number of components and row number equals nrow(datByGene) (dimension: 'annotatedFeatures x comp').

compNames: A vector of component labels with length equal to the number of component.

indComp: A vector of component indices with length equal to the number of component.

witGenes: A vector of gene IDs with length equal to the number of component.

chipManu: The manufacturer of the technology the data originates from. Useful for the annotation of the features when data originates from an _illumina_ microarray.

chipVersion: The version of the chip, only useful for when chipManu="illumina"

refSamples: A vector of sample IDs including the reference samples, e.g the "normal" samples. Must be included in sampleNames(object), i.e in colnames(dat).

typeID: A vector of characters providing the annotation IDs. It includes three elements:

geneID_annotation the IDs from the package to be used to annotate the features into genes. It will be used to fill the attributes datByGene and SByGene of the icaSet. It must match one of the objects the corresponding package supports (you can access the list of objects by typing ls("package:packagename")). If no annotation package is provided, this element is not useful.

geneID_biomart the type of gene IDs, as available in `listFilters(mart)`; where mart is specified as described in useMart. If you have directly built the IcaSet at the gene level (i.e if no annotation package is used), featureID_biomart and geneID_biomart will be identical.

featureID_biomart the type of feature IDs, as available in `listFilters(mart)`; where mart is specified as described in function useMart. Not useful if you work at the gene level.

Methods

Class-specific methods.

`getComp(IcaSet, ind, level=c("features","genes"))` Given a component index, extract the corresponding sample contribution values from A, and the feature (level="features") or gene (level="genes") projections from S. Returns a list with two elements: contrib the sample contributions and proj the feature or gene projections.

Access and set any slot specific to IcaSet:

`slotName(IcaSet)`, **and** `slotName(IcaSet)<-`: Accessing and setting any slot of name slotName contained in an IcaSet object.

`IcaSet["slotName"]`, **and** `IcaSet["slotName"]<-`: Accessing and setting any slot of name slotName contained in an IcaSet object.

Most used accessors and settors:

`A(IcaSet)`, **and** `A(IcaSet)<-`: Accessing and setting Mixing matrix A.

`S(IcaSet)`, **and** `S(IcaSet)<-`: Accessing and setting the data.frame Source S.

`Slist(IcaSet)`: Accessing the data.frame Source as a list where names are preserved.

`SByGene(IcaSet)`, **and** `SByGene(IcaSet)<-`: Accessing and setting the _annotated_ data.frame Source SByGene.

`SlistByGene(IcaSet)`: Accessing the _annotated_ Source matrix as a list where names are preserved.

`organism(IcaSet)`, `organism(IcaSet,characte)<-` Access and set value in the organism slot.

`dat(IcaSet)`, `dat(IcaSet,matrix)<-` Access and set elements named dat in the AssayData-class slot.

Derived from eSet:

`pData(IcaSet)`, `pData(IcaSet,value)<-`: See eSet

`assayData(IcaSet)`: See eSet

`sampleNames(IcaSet)` **and** `sampleNames(IcaSet)<-`: See eSet

`featureNames(IcaSet)`, `featureNames(IcaSet, value)<-`: See eSet

`dims(IcaSet)`: See eSet

`phenoData(IcaSet)`, `phenoData(IcaSet,value)<-`: See eSet

`varLabels(IcaSet)`, `varLabels(IcaSet, value)<-`: See eSet

`varMetadata(IcaSet)`, `varMetadata(IcaSet,value)<-`: See eSet

`varMetadata(IcaSet)`, `varMetadata(IcaSet,value)` See eSet

`experimentData(IcaSet)`,`experimentData(IcaSet,value)<-`: See eSet

`pubMedIds(IcaSet)`, `pubMedIds(IcaSet,value)` See eSet

`abstract(IcaSet)`: See eSet

`annotation(IcaSet)`, `annotation(IcaSet,value)<-` See eSet

`protocolData(IcaSet)`, `protocolData(IcaSet,value)<-` See eSet

`combine(IcaSet,IcaSet)`: See eSet

`storageMode(IcaSet)`, `storageMode(IcaSet,character)<-`: See eSet

Standard generic methods:

`initialize(IcaSet)`: Object instantiation, used by `new`; not to be called directly by the user.

`validObject(IcaSet)`: Validity-checking method, ensuring that `dat` is a member of `assayData`, and that the number of features, genes, samples, and components are consistent across all the attributes of the IcaSet object. `checkValidity(IcaSet)` imposes this validity check, and the validity checks of `eSet`.

`IcaSet[slotName]`, `IcaSet[slotName]<-`: Accessing and setting any slot of name `slotName` contained in an IcaSet object.

`IcaSet[i, j, k]`: Extract object of class "IcaSet" for features or genes with names i, samples with names or indices j, and components with names or indices k.

`makeDataPackage(object, author, email, packageName, packageVersion, license, biocViews, filePath`
Create a data package based on an IcaSet object. See makeDataPackage.

`show(IcaSet)`: See eSet

`dim(IcaSet)`, `ncol`: See eSet

`IcaSet[(index)]`: See eSet

`IcaSet$`, `IcaSet$<-`: See eSet

`IcaSet[[i]]`, `IcaSet[[i]]<-`: See eSet

Author(s)

Anne Biton

See Also

eSet-class, buildIcaSet, IcaSet-class, MineICAParams-class.

Examples

```
# create an instance of IcaSet
new("IcaSet")
dat <- matrix(runif(100000), nrow=1000, ncol=100)
rownames(dat) <- 1:nrow(dat)
new("IcaSet",
    dat=dat,
    A=as.data.frame(matrix(runif(1000), nrow=100, ncol=10)),
    S=as.data.frame(matrix(runif(10000), nrow=1000, ncol=10), row.names = 1:nrow(dat)))
```

`icaSetCarbayo` *IcaSet-object containing a FastICA decomposition of gene expression microarrray-based data of bladder cancer samples.*

Description

Object of class `IcaSet` containing an ICA decomposition calculated by the FastICA algorithm (through matlab function "icasso") on bladder cancer expression data measured on HG-U133A Affymetrix microarrays. The original expression data were normalized with MAS5 by the authors of the paper followed by log2-transformation. ICA was run on the dataset restricted to the 10000 most variable probe sets (based on IQR values). 10 components were computed. Only probe sets/genes having an absolute projection higher than 3 are stored in this object.

Author(s)

Anne Biton

References

http://jco.ascopubs.org/content/24/5/778/suppl/DC1

`icaSetKim` *IcaSet-object containing a FastICA decomposition of gene expression microarrray-based data of bladder cancer samples.*

Description

Object of class `IcaSet` containing an ICA decomposition calculated by the FastICA algorithm (through matlab function "icasso") on bladder cancer expression data measured on illumina Human-6 BeadChip, version 2. It contains 20 independent components. The original expression data contain 165 tumor samples, were normalized by the authors of the paper with Illumina BeadStudio software using Quantile normalization and log2 transformation, and are restricted to the 10000 most variable probe sets.

Author(s)

Anne

References

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE13507

`icaSetRiester` *IcaSet-object containing a FastICA decomposition of gene expression microarrray-based data of bladder cancer samples.*

Description

Object of class `IcaSet` containing an ICA decomposition calculated by the FastICA algorithm (through matlab function "icasso") on gene expression data of urothelial tumors. measured on a HG-U133-plus2 Affymetrix microarrays. It contains 20 independent components. The original expression data contain 93 tumor samples, were normalized with GCRMA with log2-transformation, and are restricted to the 10000 most variable probe sets.

Author(s)

Anne Biton

References

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE31684

`icaSetStransky` *IcaSet-object containing a FastICA decomposition of gene expression microarrray-based data of bladder cancer samples.*

Description

Object of class `IcaSet` containing an ICA decomposition calculated by the FastICA algorithm (through matlab function "icasso") on bladder cancer expression data measured on HG-U133-95a and HG-U133-95av2 Affymetrix microarrays. It contains 20 independent components. The original expression data contain 63 tumor samples and were normalized by RMA with log2-transformation.

Author(s)

Anne Biton

References

http://microarrays.curie.fr/publications/oncologie_moleculaire/bladder_TCM/

indComp — *Retrieve and set component labels, indices, and witness genes from IcaSet*

Description

These generic functions access and set the attributes `compNames`, `indComp` and `witGenes` stored in an object of class `IcaSet`.

Usage

```
indComp(object)
indComp(object) <- value
compNames(object)
compNames(object) <- value
witGenes(object)
witGenes(object) <- value
```

Arguments

`object`	object of class `IcaSet`
`value`	Numeric vector for `indComp`, character vector for `compNames` and `witGenes`, with length equal to `ncol(A(object))` and containing: component indices (for `indComp`), labels (for `compNames`), or gene witness IDs (for `witGenes`).

Value

`indComp` returns a numeric vector containing component indices; `compNames` returns a character vector containing component labels; `witGenes` returns a character vector containing witness genes IDs.

Author(s)

Anne Biton

MineICAParams — *Class to contain parameters for the analysis of an ICA decomposition.*

Description

Container for parameters used during the analysis of an ICA decomposition obtained on genomics data.

Creating Objects

```
new("MineICAParams")
new("MineICAParams", resPath="", genesPath="ProjByComp", pvalCutoff=0.05, selCutoff=3)
```

Slots

`Sfile` A txt file containing the Source matrix S.

`Afile` A txt file containing the Mixing matrix A.

`datfile` A txt file containing the data (typically expression data) on which the decomposition was calculated.

`annotfile` Either a RData or txt file containing the annotation data for the samples (must be of dimensions samples*annotations).

`resPath` The path where the outputs of the analysis will be written.

`genesPath` The path _within_ the resPath where the gene projections will be written. If missing, will be automatically attributed as resPath/gene2components/.

`annot2col` A vector of colors indexed by annotation levels. If missing, will be automatically attributed using function `annot2Color`.

`pvalCutoff` The cutoff used to consider a p-value significant, default is 0.05.

`selCutoff` The cutoff applied on the absolute feature/gene projection values to consider gene as contributing to a component, default is 3. Must be either of length 1 and the same treshold is applied to all components, or of length equal to the number of components in order to use a specific threshold for each component.

Methods

For any slot:

Accessing and setting any slot of name `slotName` contained in an MineICAParams object.

`slotName(MineICAParams)` **and** `slotName(MineICAParams) <-` **and** `MineICAParams["slotName"]` **and** `MineICAParams["sl`
Accessing and setting any slot of name `slotName` contained in an MineICAParams object.

Author(s)

Anne Biton

See Also

`MineICAParams-class`, `runAn`.

Examples

```
# create an instance of LocSet
new("MineICAParams")
```

nbOccByGeneInComp *nbOccByGeneInComp*

Description

For each feature/gene, this function returns the indices of the components they contribute to.

Usage

```
nbOccByGeneInComp(Slist, cutoff, sel)
```

Arguments

Slist	A list whose each element contains projection values of features/genes on a component.
cutoff	A threshold to be used to define a gene as contributor
sel	A list whose each element contains projection values of contributing features/genes on a component (the difference with arg Slist is that sel is already restricted to the contributing genes).

Value

This function returns a list which gives for each feature/gene the indices of the components it contributes to.

Author(s)

Anne Biton

Examples

```
c1 <- rnorm(100); names(c1) <- paste("g",100:199,sep="")
c2 <- rnorm(100); names(c2) <- paste("g",1:99,sep="")
MineICA:::nbOccByGeneInComp(Slist=list(c1,c2), cutoff= 0.5)
```

nbOccInComp *Select components the features contribute to*

Description

For each feature/gene, this function returns the components they contribute to and their projection values across all the components.

Usage

```
nbOccInComp(icaSet, params, selectionByComp = NULL,
  level = c("features", "genes"), file = NULL)
```

Arguments

icaSet	An object of class IcaSet
params	An object of class MineICAParams containing the parameters of the analysis, the attribute cutoffSel is used as a threshold on the absolute projections to determine which genes contribute to the components.
selectionByComp	The list of components already restricted to the contributing genes
level	The attribute of icaSet to be used, are reported the occurences of either the "features" or the "genes".
file	The file where the output data.frame and plots are written.

Details

A feature/gene is considered as a contributor when its scaled projection value exceeds the threshold selCutoff(icaSet).

This function plots the number of times the feature/gene is a contributor as a function of the standard deviation of its expression profile.

The created files are located in genePath(params). An extensiom '.htm' and '.pdf' is respectively added to the file name for the data.frame and the plot outputs.

Value

Returns a data.frame whose columns are: 'gene' the feature or gene ID, 'nbOcc' the number of components on which the gene contributes according to the threshold, 'components' the indices of these components, and then the component indices which contain its projection values.

Author(s)

Anne Biton

Examples

```
data(icaSetCarbayo)
params <- buildMineICAParams(resPath="carbayo/")
nbOcc <- nbOccInComp(icaSet=icaSetCarbayo, params=params, level="genes", file="gene2MixingMatrix")
```

---

nodeAttrs *Generate node attributes*

Description

This function builds a data.frame describing for each node of the graph its ID and which analysis/data it comes from.

Usage

```
nodeAttrs(nbAn, nbComp, labAn, labComp, file)
```

Arguments

nbAn	Number of analyses being considered, i.e number of IcaSet objects
nbComp	Number of components by analysis, if of length 1 then it is assumed that each analysis has the same number of components.
labAn	Labels of the analysis, if missing it will be generated as an1, an2, ...
labComp	List containing the component labels indexed by analysis, if missing will be generated as comp1, comp2, ...
file	File where the description of the node attributes will be written

Details

The created file is used in Cytoscape.

Value

A data.frame describing each node/component

Author(s)

Anne Biton

Examples

```
## 4 datasets, 20 components calculated in each dataset, labAn
nodeAttrs(nbAn=4, nbComp=20, labAn=c("tutu","titi","toto","tata"))
```

---

plotAllMix *Plots the Gaussian fitted by* Mclust *on several numeric vectors*

Description

Given a result of function Mclust applied on several numeric vectors, this function plots the fitted Gaussian on their histograms.

Usage

```
plotAllMix(mc, A, nbMix = NULL, pdf, nbBreaks = 20,
  xlim = NULL)
```

Arguments

mc	A list consisting of outputs of function Mclust applied to each column of A, if this argument is missing Mclust is applied by the function.
A	A data.frame of dimensions 'samples x components'.
nbMix	The number of Gaussian to be fitted.
nbBreaks	The number of breaks for the histogram.
xlim	x-axis limits to be used in the plot.
pdf	A pdf file.

Details

This function can only deal with at the most three Gaussian

Value

A list of `Mclust` results.

Author(s)

Anne Biton

See Also

`plotMix`, `hist`, `Mclust`

Examples

```
A <-matrix(c(c(rnorm(80,mean=-0.5,sd=1),rnorm(80,mean=1,sd=0.2)),rnorm(160,mean=0.5,sd=1),
          c(rnorm(80,mean=-1,sd=0.3),rnorm(80,mean=0,sd=0.2))),ncol=3)
## apply function Mclust to each column of A
mc <- apply(A,2,Mclust)
## plot the corresponding Gaussians on the histogram of each column
plotAllMix(mc=mc,A=A)
## apply function Mclust to each column of A, and impose the fit of two Gaussian (G=2)
mc <- apply(A,2,Mclust,G=2)
## plot the corresponding Gaussians on the histogram of each column
plotAllMix(mc=mc,A=A)
## When arg 'mc' is missing, Mclust is applied by the function
plotAllMix(A=A)
```

---

plotCorGraph *Plots graph using*

Description

This function plots the correlation graph in an interactive device using function `tkplot`.

Usage

```
plotCorGraph(dataGraph, edgeWeight = "cor", nodeAttrs,
  nodeShape, nodeCol = "labAn", nodeName = "indComp",
  col, shape, title = "", reciproCol = "reciprocal",
  tkplot = FALSE, ...)
```

Arguments

`dataGraph`	A data.frame containing the graph description. It must have two columns `n1` and `n2`, each row denoting that there is an edge from `n1` to `n2`. Node labels in columns `n1` and `n2` of `dataGraph` must correspond to node IDs in column `id` of `nodeAttrs`.
`edgeWeight`	The column of dataGraph used to weight edges.
`nodeAttrs`	A data.frame with node description, see function `nodeAttrs`.

nodeShape	Denotes the column of nodeAttrs used to attribute the node shapes.
nodeCol	Denotes the column of nodeAttrs used to color the nodes in the graph.
nodeName	Denotes the column of nodeAttrs used as labels for the nodes in the graph.
col	A vector of colors, for the nodes, indexed by the unique elements of nodeCol column from nodeAttrs. If missing, colors will be automatically attributed.
shape	A vector of shapes indexed by the unique elements of column nodeShape from nodeAttrs. If missing, shapes will be automatically attributed.
title	Title for the plot
reciproCol	Denotes the column of dataGraph containing TRUE if the row defines a reciprocal node, else FALSE. See annotReciprocal.
tkplot	If TRUE, performs interactive plot with function tkplot, else uses plot.igraph.
...	Additional parameters as required by tkplot.

Details

You have to slighly move the nodes to see cliques because strongly related nodes are often superimposed. The edgeWeight column is used to weight the edges within the fruchterman.reingold layout available in the package igraph.

The argument nodeCol typically denotes the column containing the names of the datasets. Colors are automatically attributed to the nodes using palette Set3 of package RColorBrewer. The corresponding colors can be directly specified in the 'col' argument. In that case, 'col' must be a vector of colors indexed by the unique elements contained in nodeCol column (e.g dataset ids).

As for colors, one can define the column of nodeAttrs that is used to define the node shapes. The corresponding shapes can be directly specified in the shape argument. In that case, shape must be one of c("circle","square", " vcsquare", "rectangle", "crectangle", "vrectangle") and must be indexed by the unique elements of nodeShape column.

Unfortunately, shapes can't be taken into account when tkplot is TRUE (interactive plot).

If reciproCol is not missing, it is used to color the edges, either in grey if the edge is not reciprocal or in black if the edge is reciprocal.

Value

A list consisting of

dataGraph a data.frame defining the correlation graph

nodeAttrs a data.frame describing the node of the graph

graph the graph as an object of class igraph

graphid the id of the graph plotted using tkplot

Author(s)

Anne Biton

See Also

compareAn, nodeAttrs, compareAn2graphfile, runCompareIcaSets

Examples

```
dat1 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat1) <- paste("g", 1:1000, sep="")
colnames(dat1) <- paste("s", 1:10, sep="")
dat2 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat2) <- paste("g", 1:1000, sep="")
colnames(dat2) <- paste("s", 1:10, sep="")

## run ICA
resJade1 <- runICA(X=dat1, nbComp=3, method = "JADE")
resJade2 <- runICA(X=dat2, nbComp=3, method = "JADE")

## build params
params <- buildMineICAParams(resPath="toy/")

## build IcaSet object
icaSettoy1 <- buildIcaSet(params=params, A=data.frame(resJade1$A), S=data.frame(resJade1$S),
                          dat=dat1, alreadyAnnot=TRUE)$icaSet
icaSettoy2 <- buildIcaSet(params=params, A=data.frame(resJade2$A), S=data.frame(resJade2$S),
                          dat=dat2, alreadyAnnot=TRUE)$icaSet
icaSets <- list(icaSettoy1, icaSettoy2)

resCompareAn <- compareAn(icaSets=list(icaSettoy1,icaSettoy2), labAn=c("toy1","toy2"),
                         type.corr="pearson", level="genes", cutoff_zval=0)

## Build a graph where edges correspond to maximal correlation value (useVal="cor"),
dataGraph <- compareAn2graphfile(listPairCor=resCompareAn, useMax=TRUE, useVal="cor", file="myGraph.txt")

## construction of the data.frame with the node description
nbComp <- rep(3,2) #each IcaSet contains 3 components
nbAn <- 2 # we are comparing 2 IcaSets
# labels of components created as comp*i*
labComp <- foreach(icaSet=icaSets, nb=nbComp, an=1:nbAn) %do% {
                  paste(rep("comp",sum(nb)),1:nbComp(icaSet),sep = "")}

# creation of the data.frame with the node description
nodeDescr <- nodeAttrs(nbAn = nbAn, nbComp = nbComp, labComp = labComp,
                       labAn = c("toy1","toy2"), file = "nodeInfo.txt")

## Plot correlation graph, slightly move the attached nodes to make the cliques visible
## use tkplot=TRUE to have an interactive graph
res <- plotCorGraph(title = "Compare toy 1 and 2", dataGraph = dataGraph, nodeName = "indComp", tkplot = FALSE
             nodeAttrs = nodeDescr, edgeWeight = "cor", nodeShape = "labAn", reciproCol = "reciprocal")


## Not run:
## load two microarray datasets
library(breastCancerMAINZ)
library(breastCancerVDX)
data(mainz)
data(vdx)

## Define a function used to build two examples of IcaSet objects
treat <- function(es, annot="hgu133a.db") {
   es <- selectFeatures_IQR(es,10000)
   exprs(es) <- t(apply(exprs(es),1,scale,scale=FALSE))
```

```
   colnames(exprs(es)) <- sampleNames(es)
   resJade <- runICA(X=exprs(es), nbComp=10, method = "JADE", maxit=10000)
  resBuild <- buildIcaSet(params=buildMineICAParams(), A=data.frame(resJade$A), S=data.frame(resJade$S),
                      dat=exprs(es), pData=pData(es), refSamples=character(0),
                      annotation=annot, typeID= typeIDmainz,
                      chipManu = "affymetrix", mart=mart)
   icaSet <- resBuild$icaSet
}
## Build the two IcaSet objects
icaSetMainz <- treat(mainz)
icaSetVdx <- treat(vdx)

icaSets <- list(icaSetMainz, icaSetVdx)
labAn <- c("Mainz", "Vdx")

## correlations between gene projections of each pair of IcaSet
resCompareAn <- compareAn(icaSets = icaSets, level = "genes", type.corr= "pearson",
                          labAn = labAn, cutoff_zval=0)

## construction of the correlation graph using previous output
dataGraph <- compareAn2graphfile(listPairCor=resCompareAn, useMax=TRUE, file="corGraph.txt")

## construction of the data.frame with the node description
nbComp <- rep(10,2) #each IcaSet contains 10 components
nbAn <- 2 # we are comparing 2 IcaSets
# labels of components created as comp*i*
labComp <- foreach(icaSet=icaSets, nb=nbComp, an=1:nbAn) %do% {
                 paste(rep("comp",sum(nb)),1:nbComp(icaSet),sep = "")}

# creation of the data.frame with the node description
nodeDescr <- nodeAttrs(nbAn = nbAn, nbComp = nbComp, labComp = labComp,
    labAn = labAn, file = "nodeInfo.txt")

## Plot correlation graph, slightly move the attached nodes to make the cliques visible
res <- plotCorGraph(title = "Compare two ICA decomsitions obtained on \n two
             microarray-based data of breast tumors", dataGraph = dataGraph, nodeName = "indComp",
             nodeAttrs = nodeDescr, edgeWeight = "cor", nodeShape = "labAn", reciproCol = "reciprocal")


## End(Not run)
```

`plotMix` *Plots an histogram and Gaussian fitted by* `Mclust`

Description

Given a result of function `Mclust` applied to a numeric vector, this function draws the fitted Gaussian on the histogram of the data values.

Usage

```
plotMix(mc, data, nbBreaks, traceDensity = TRUE,
  title = "", xlim, ylim, ...)
```

Arguments

mc	The result of Mclust function applied to argument data
data	A vector of numeric values
nbBreaks	The number of breaks for the histogram
traceDensity	If TRUE (default) density are displayed on the y-axis, else if FALSE counts are displayed on the y-acis
title	A title for the plot
xlim	x-axis limits to be used in the plot
ylim	y-axis limits to be used in the plot
...	additional arguments for hist

Details

A shapiro test p-value is added to the plot title. This function can only deal with at the most three Gaussian.

Value

NULL

Author(s)

Anne Biton

See Also

hist, Mclust

Examples

```
## create a mix of two Gaussian
v <-c(rnorm(80,mean=-0.5,sd=1),rnorm(80,mean=1,sd=0.2))
## apply Mclust
mc <- Mclust(v)
## plot fitted Gaussian on histogram of v
plotMix(mc=mc,data=v,nbBreaks=30)
```

---

plotPosAnnotInComp *Histograms of sample contributions for each annotation level*

Description

This function plots the positions of groups of samples formed by the variables (i.e the sample annotations) across all the components of an object of class icaSet. For each variable level (e.g for each tumor stage) this function plots the positions of the corresponding samples (e.g the subset of samples having this tumor stage) within the histogram of the global sample contributions. The plots are saved in pdf file, one file is created per variable. The pdf files are names 'variable.pdf' and save either in pathPlot if specified or the current directory.

Usage

```
plotPosAnnotInComp(icaSet, params,
  keepVar = varLabels(icaSet),
  keepComp = indComp(icaSet),
  keepSamples = sampleNames(icaSet), pathPlot = NULL,
  breaks = 20, colAll = "grey74", colSel, resClus,
  funClus = c("Mclust", "kmeans"), nbClus = 2,
  by = c("annot", "component"),
  typeImage = c("pdf", "png", "none"), ...)
```

Arguments

icaSet	An object of class IcaSet
params	A MineICAParams object
keepVar	The variable labels to be considered, i.e a subset of the column labels of the pheno data of icaSet available in (varLabels(icaSet))
keepComp	A subset of components available in indComp(icaSet); by default, all components are used
keepSamples	A subset of samples, must be available in sampleNames(icaSet); by default, all samples are used
pathPlot	A character specifying the path where the plots will be saved
breaks	The number of breaks to be used in the histograms
colSel	The colour of the histogram of the group of interest, default is "red"
colAll	The colour of the global histogram, default is "grey74"
resClus	A list containing the outputs of function clusterSamplesByComp, which consists of sample clustering applied to matrix A of argument icaSet. If missing, the clustering is performed by the function.
funClus	The clustering method to be used, either "Mclust" or "kmeans". If resClus is not missing, equals resClus$funClus.
nbClus	If resClus is missing, it provides the number of clusters to be computed by funClus, default is 2
by	Either "annot" to plot the histograms of each variable across all components, or "component" to plot the histograms for each component across variables. When by="annot" one pdf file is created by variable name, while when annot="component", one pdf file is created by component.
typeImage	The type of image to be created, either "pdf" (default) or "png". "png" is not recommended, unless there are at the most 4 histograms to be plotted, because it does not allow to deal with multiple pages of plots.
...	Additional parameters for function hist

Details

The plotted values are the sample contributions across the components, i.e across the columns of A(icaSet).

If argument resClus is missing, the function computes the clustering of the samples on each component (i.e on each column of A(icaSet)) using funClus and nbClus.

The association between the clusters and the considered sample group is tested using a chi-square test. The p-values of these tests are available in the title of each plot.

When by="annot" this function plots the histograms of each variable across all components, to plot the histograms for each component across variables, please use by="component".

Value

NULL

Author(s)

Anne Biton

See Also

plotPosSamplesInComp, chisq.test

Examples

```
## Not run:
## load an example of IcaSet
data(icaSetCarbayo)

## Use icaSetCarbayo, look at the available annotations
varLabels(icaSetCarbayo)

## Plot positions of samples in components according to annotations 'SEX' and 'STAGE'
# plots are saved in files SEX.pdf and STAGE.pdf created in the current directory
plotPosAnnotInComp(icaSet=icaSetCarbayo, keepVar=c("SEX","STAGE"), keepComp=1:2,  funClus="Mclust")
# specifiy arg 'pathPlot' to save the pdf in another directory, but make sure it exists before
# specifiy arg 'by="comp"' to create one pdf file per component

## End(Not run)
```

---

plot_heatmapsOnSel *Plot heatmap associated with each component*

Description

This function plots the heatmaps representing the measured values of the contributing features/genes on each component. It also plots the sample annotations above each heatmap using colours.

Usage

```
plot_heatmapsOnSel(icaSet, selCutoff = 4,
  level = c("features", "genes"), samplesOrder,
  featuresOrder, selectionByComp, keepVar,
  keepComp = indComp(icaSet), doSamplesDendro = TRUE,
  doGenesDendro = TRUE,
  heatmapCol = maPalette(low = "blue", high = "red", mid = "yellow", k = 44),
  file = "", path = "", annot2col, ...)
```

Arguments

icaSet	The IcaSet object
selCutoff	A numeric threshold used to select the contributing genes based on their projection values. Must be either of length 1 and the same treshold is applied to all components, or of length equal to the number of components and one specific threshold is used for each component.
samplesOrder	A list providing the order of the samples, per component, to be used in the heatmaps. If missing, the contribution values of the samples are used to rank the columns of the heatmaps.
featuresOrder	A list providing the order of the genes, per component, to be used in the heatmaps. If missing, the projection values of the genes are used to rank the rows of the heatmaps.
selectionByComp	A list of gene projections per component already restricted to the contributing genes, if missing is computed by the function.
level	A character indicating which data level is used to plot the heatmaps: either 'features' to represent the data at the feature levels (e.g expression profiles of probe sets), or 'genes' to represent the data at the annotated-features level (e.g gene expression profiles).
keepVar	The variable labels to be considered, i.e a subset of the column labels of the pheno data of icaSet available in (varLabels(icaSet))
keepComp	A subset of components, must be included in indComp(icaSet). By default, all components are used.
doSamplesDendro	A logical indicating whether a hierarchical clustering has to be performed on the data matrix restricted to the contributing features/genes, and whether the corresponding dendrogram has to be plotted, default is TRUE.
doGenesDendro	A logical indicating if the dendrogram of features/genes has to be plotted, default is FALSE.
heatmapCol	A list of colors used to for heatmap coloring (see argument col of the function image).
file	A character to add to each pdf file name. This function creates one file by component named "index-of-component_file.pdf" .
path	A directory for the output pdf files, must end with "/". Default is current directory.
annot2col	A vector of colours indexed by the levels of the variables of icaSet (i.e all the annotation values available in pData(icaSet)). If missing the colours are generated automatically using the function annot2Color
...	Additional parameters for function heatmap.plus

Details

This function restricts the data matrix of an IcaSet object to the contributing genes/features, and order features/genes and samples either as asked by the user or according to their values in the ICA decomposition.

The heatmap is plotted using a slightly modified version of the function heatmap.plus from the package of the same name. By default in this function, the hierarchical clustering is calculated using the function agnes with euclidean metric and Ward's method.

Value

A list with one element per component, each of them being a list consisting of three elements:

x the matrix represented by the heatmap,
breaks the breaks used for the colours of the heatmap,
dendro the dendrogram.

Author(s)

Anne Biton

See Also

heatmap.plus, image, annot2Color, build_sortHeatmap

Examples

```
## Not run:
## load an example of IcaSet object
data(icaSetCarbayo)

## check which variables you would like to use in the heatmap
varLabels(icaSetCarbayo)
keepVar <- c("STAGE","SEX")
## Use only component 1
keepComp <- 1

## For each component, select contributing *genes* using a threshold of 2 on the absolute projection values,
## and plot heatmaps of these contributing genes by ordering genes and samples according to their contributior
plot_heatmapsOnSel(icaSet = icaSetCarbayo, selCutoff = 2, level = "genes", keepVar = keepVar,
                   keepComp=1, doSamplesDendro = TRUE, doGenesDendro = TRUE,
                 heatmapCol = maPalette(low = "blue",high = "red", mid = "yellow", k=44),
                   file = "heatmapWithoutDendro_zval3.pdf")

## For each considered component, select contributing *features* using a threshold of 2 on the absolute projec
## and plot heatmaps of these contributing genes with dendrograms
plot_heatmapsOnSel(icaSet = icaSetCarbayo, selCutoff = 2, level = "features", keepVar = keepVar,
                   keepComp=1, doSamplesDendro = TRUE, doGenesDendro = TRUE,
                 heatmapCol = maPalette(low = "blue",high = "red", mid = "yellow", k=44),
                   file = "heatmapWithDendro_zval3.pdf")




## End(Not run)
```

---

qualVarAnalysis *Tests association between qualitative variables and components.*

Description

This function tests if the groups of samples formed by the variables are differently distributed on the components, in terms of contribution value (i.e of values in matrix A(icaSet)). The distribution of the samples on the components are represented using either density plots of boxplots. It is possible to restrict the tests and the plots to a subset of samples and/or components.

Usage

```
qualVarAnalysis(params, icaSet, keepVar,
  keepComp = indComp(icaSet),
  keepSamples = sampleNames(icaSet),
  adjustBy = c("none", "component", "variable"),
  method = "BH", doPlot = TRUE, typePlot = "density",
  addPoints = FALSE, onlySign = TRUE,
  cutoff = params["pvalCutoff"],
  colours = annot2col(params), path = "qualVarAnalysis/",
  filename = "qualVar", typeImage = "png")
```

Arguments

params	An object of class MineICAParams providing the parameters of the analysis.
icaSet	An object of class IcaSet.
keepVar	The variable labels to be considered, must be a subset of varLabels(icaSet).
keepComp	A subset of components, must be included in indComp(icaSet). By default, all components are used.
keepSamples	A subset of samples, must be included in sampleNames(icaSet). By default, all samples are used.
adjustBy	The way the p-values of the Wilcoxon and Kruskal-Wallis tests should be corrected for multiple testing: "none" if no p-value correction has to be done, "component" if the p-values have to be corrected by component, "variable" if the p-values have to be corrected by variable
method	The correction method, see p.adjust for details, default is "BH" for Benjamini & Hochberg.
doPlot	If TRUE (default), the plots are done, else only tests are performed.
addPoints	If TRUE, points are superimposed on the boxplot.
typePlot	The type of plot, either "density" or "boxplot".
onlySign	If TRUE (default), only the significant results are plotted.
cutoff	A threshold p-value for statistical significance.
colours	A vector of colours indexed by the variable levels, if missing the colours are automatically generated using annot2Color.
path	A directory _within resPath(params)_ where the files containing the plots and the p-value results will be located. Default is "qualVarAnalysis/".
typeImage	The type of image file to be used.
filename	The name of the HTML file containing the p-values of the tests, if NULL no file is created.

Details

This function writes an HTML file containing the results of the tests as a an array of dimensions 'variables * components' containing the p-values of the tests. When a p-value is considered as significant according to the threshold cutoff, it is written in bold and filled with a link pointing to the corresponding plot. One image is created by plot and located into the sub-directory "plots/" of path. Each image is named by index-of-component_var.png. Wilcoxon or Kruskal-Wallis tests are performed depending on the number of groups of interest in the considered variable (argument keepLev).

Value

Returns A data.frame of dimensions 'components x variables' containing the p-values of the non-parametric tests (Wilcoxon or Kruskal-Wallis tests) wich test if the samples groups defined by each variable are differently distributed on the components.

Author(s)

Anne Biton

See Also

, qualVarAnalysis, p.adjust, link{writeHtmlResTestsByAnnot}, wilcox.test, kruskal.test

Examples

```
## load an example of IcaSet
data(icaSetCarbayo)

## build MineICAParams object
params <- buildMineICAParams(resPath="carbayo/")

## Define the directory containing the results
dir <- paste(resPath(params), "comp2annot/", sep="")

## Run tests, make no adjustment of the p-values,
# for variable grade and components 1 and 2,
# and plot boxplots when 'doPlot=TRUE'.
qualVarAnalysis(params=params, icaSet=icaSetCarbayo, adjustBy="none", typePlot="boxplot",
                keepVar="GRADE", keepComp=1:2, path=dir, doPlot=FALSE)
```

---

quantVarAnalysis *Correlation between variables and components.*

Description

This function tests if numeric variables are correlated with components.

Usage

```
quantVarAnalysis(params, icaSet, keepVar,
  keepComp = indComp(icaSet),
  keepSamples = sampleNames(icaSet),
  adjustBy = c("none", "component", "variable"),
  method = "BH", typeCor = "pearson", doPlot = TRUE,
  onlySign = TRUE, cutoff = 0.4,
  cutoffOn = c("cor", "pval"), colours,
  path = "quantVarAnalysis/", filename = "quantVar",
  typeImage = "png")
```

Arguments

params	An object of class MineICAParams providing the parameters of the analysis.
icaSet	An object of class IcaSet.
keepVar	The variable labels to be considered, must be a subset of varLabels(icaSet).
keepComp	A subset of components, must be included in indComp(icaSet). By default, all components are used.
keepSamples	A subset of samples, must be included in sampleNames(icaSet). By default, all samples are used.
adjustBy	The way the p-values of the Wilcoxon and Kruskal-Wallis tests should be corrected for multiple testing: "none" if no p-value correction has to be done, "component" if the p-values have to be corrected by component, "variable" if the p-values have to be corrected by variable
method	The correction method, see p.adjust for details, default is "BH" for Benjamini & Hochberg.
doPlot	If TRUE (default), the plots are done, else only tests are performed.
onlySign	If TRUE (default), only the significant results are plotted.
cutoff	A threshold p-value for statistical significance.
cutoffOn	The value the cutoff is applied to, either "cor" for correlation or "pval" for p-value
typeCor	the type of correlation to be used, one of c("pearson","spearman","kendall").
colours	A vector of colours indexed by the variable levels, if missing the colours are automatically generated using annot2Color.
path	A directory _within resPath(params)_ where the files containing the plots and the p-value results will be located. Default is "quantVarAnalysis/".
typeImage	The type of image file to be used.
filename	The name of the HTML file containing the p-values of the tests, if NULL no file is created.

Details

This function writes an HTML file containing the correlation values and test p-values as a an array of dimensions 'variables * components' containing the p-values of the tests. When a p-value is considered as significant according to the threshold cutoff, it is written in bold and filled with a link pointing to the corresponding plot. One image is created by plot and located into the sub-directory "plots/" of path. Each image is named by index-of-component_var.png.

Value

Returns A data.frame of dimensions 'components x variables' containing the p-values of the non-parametric tests (Wilcoxon or Kruskal-Wallis tests) wich test if the samples groups defined by each variable are differently distributed on the components.

Author(s)

Anne Biton

See Also

qualVarAnalysis, p.adjust, link{writeHtmlResTestsByAnnot}, code

Examples

```
## load an example of IcaSet
data(icaSetCarbayo)

# build MineICAParams object
params <- buildMineICAParams(resPath="carbayo/")

# Define the directory containing the results
dir <- paste(resPath(params), "comp2annottest/", sep="")

# Check which variables are numeric looking at the pheno data, here only one  -> AGE
# pData(icaSetCarbayo)

## Perform pearson correlation tests and plots association corresponding
# to correlation values larger than 0.2
quantVarAnalysis(params=params, icaSet=icaSetCarbayo, keepVar="AGE", keepComp=1:2,
                 adjustBy="none", path=dir, cutoff=0.2, cutoffOn="cor")

## Not run:
## Perform Spearman correlation tests and do scatter plots for all pairs
quantVarAnalysis(params=params, icaSet=icaSetCarbayo, keepVar="AGE", adjustBy="none", path=dir,
                 cutoff=0.1, cutoffOn="cor", typeCor="spearman", onlySign=FALSE)

## Perform pearson correlation tests and plots association corresponding
# to p-values lower than 0.05 when 'doPlot=TRUE'
quantVarAnalysis(params=params, icaSet=icaSetCarbayo, keepVar="AGE", adjustBy="none", path=dir,
                 cutoff=0.05, cutoffOn="pval", doPlot=FALSE)

## End(Not run)
```

`relativePath` *Relative path*

Description

Computes the relative path between two imbricated paths

Usage

```
relativePath(path1, path2)
```

Arguments

`path1`	The first path
`path2`	The second path

Details

`path1` and `path2` must be imbricated.

Value

The relative path between path1 and path2

Author(s)

Anne Biton

Examples

```
path1 <- "home/lulu/res/gene2comp/"
path2 <- "home/lulu/res/comp2annot/invasive/"
relativePath(path1,path2)
```

---

runAn *Run analysis of an IcaSet object*

Description

This function runs the analysis of an ICA decomposition contained in an IcaSet object, according to the parameters entered by the user and contained in a MineICAParams.

Usage

```
runAn(params, icaSet, keepVar,
  heatmapCutoff = params["selCutoff"],
  funClus = c("Mclust", "kmeans"), nbClus,
  clusterOn = "A", keepComp, keepSamples,
  adjustBy = c("none", "component", "variable"),
  typePlot = c("boxplot", "density"),
  mart = useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl"),
  dbGOstats = c("KEGG", "GO"), ontoGOstats = "BP",
  condGOstats = TRUE,
  cutoffGOstats = params["pvalCutoff"],
  writeGenesByComp = TRUE, writeFeaturesByComp = FALSE,
  selCutoffWrite = 2.5, runVarAnalysis = TRUE,
  onlySign = T, runClustering = FALSE, runGOstats = TRUE,
  plotHist = TRUE, plotHeatmap = TRUE)
```

Arguments

params	An object of class MineICAParams containing the parameters of the analysis.
icaSet	An object of class IcaSet.
keepVar	The variable labels to be considered, i.e a subset of the annotation variables available in (varLabels(icaSet)).
keepSamples	The samples to be considered, i.e a subset of (sampleNames(icaSet)).
heatmapCutoff	The cutoff (applied to the scaled feature/gene projections contained in S/SByGene) used to select the contributing features/genes.
funClus	The function to be used to cluster the samples, must be one of c("Mclust","kmeans","pam","pamk" Default is "Mclust".
nbClus	The number of clusters to be computed when applying funClus. Can be missing (default) if funClus="Mclust" or funClus="pamk".
keepComp	The indices of the components to be analyzed, must be included in indComp(icaSet). If missing, all components are treated.

`adjustBy` The way the p-values of the Wilcoxon and Kruskal-Wallis tests should be corrected for multiple testing: `"none"` if no p-value correction has to be done, `"component"` if the p-values have to be corrected by component, `"annotation"` if the p-values have to be corrected by variable

`typePlot` The type of plot used to show distribution of sample-groups contributions, either "density" or "boxplot"

`mart` A mart object used for annotation, see function `useMart`

`dbGOstats` The used database to use ('GO' and/or 'KEGG'), default is both.

`ontoGOstats` A string specifying the GO ontology to use. Must be one of 'BP', 'CC', or 'MF', see `GOHyperGParams`. Only used when argument `dbGOstats` is 'GO'.

`condGOstats` A logical indicating whether the calculation should conditioned on the GO structure, see `GOHyperGParams`.

`cutoffGOstats` The p-value threshold used for selecting enriched gene sets, default is params["pvalCutoff"]

`writeGenesByComp`
If TRUE (default) the gene projections (`SByGene(icaSet)`) are written in an html file and annotated using `biomaRt` for each component.

`writeFeaturesByComp`
If TRUE (default) the feature projections (`S(icaSet)`) are written in an html file and annotated using `biomaRt` for each component.

`runGOstats` If TRUE the enrichment analysis of the contributing genes is run for each component using package `GOstats` (default is TRUE).

`plotHist` If TRUE the position of the sample annotations within the histograms of the sample contributions are plotted.

`plotHeatmap` If TRUE the heatmap of the contributing features/genes are plotted for each component.

`runClustering` If TRUE the potential associations between a clustering of the samples (performed according to the components), and the sample annotations, are tested using chi-squared tests.

`runVarAnalysis` If TRUE the potential associations between sample contributions (contained in `A(icaSet)`) are tested using Wilcoxon or Kruskal-Wallis tests.

`onlySign` If TRUE (default), only the significant results are plotted in functions `qualVarAnalysis, quantVar`, else all plots are done.

`selCutoffWrite` The cutoff applied to the absolute feature/gene projection values to select the features/genes that will be annotated using package `biomaRt`, default is 2.5.

`clusterOn` Specifies the matrix used to apply clustering if `runClustering=TRUE`:

- `"A"`**:** the clustering is performed in one dimension, on the vector of sample contributions,
- **"S":** the clustering is performed on the original data restricted to the contributing individuals,
- **"AS":** the clustering is performed on the matrix formed by the product of the column of A and the row of S.

Details

This function calls functions of the MineICA package depending on the arguments:

`writeProjByComp` **(if** `writeGenesByComp=TRUE` **or** `writeFeaturesByComp`**)** which writes in html files the description of the features/genes contributing to each component, and their projection values on all the components.

`plot_heatmapsOnSel` **(if** `plotHeatmap=TRUE`**)** which plots heatmaps of the data restricted to the contributing features/genes of each component.

`plotPosAnnotInComp` **(if** `plotHist=TRUE`**)** which plots, within the histogram of the sample contribution values of every component, the position of groups of samples formed according to the sample annotations contained in `pData(icaSet)`.

`clusterSamplesByComp` **(if** `runClustering=TRUE`**)** which clusters the samples according to each component.

`clusVarAnalysis` **(if** `runClustering=TRUE`**)** which computes the chi-squared test of association between a given clustering of the samples and each annotation level contained in `pData(icaSet)`, and summarizes the results in an HTML file.

`runEnrich` **(if** `runGOstats=TRUE`**)** which performs enrichment analysis of the contributing genes of the components using package GOstats.

`qualVarAnalysis` **and** `quantVarAnalysis` **(if** `varAnalysis=TRUE`**)** which tests if the groups of samples formed according to sample annotations contained in `pData(icaSet)` are differently distributed on the components, in terms of contribution value.

Several directories containing the results of each analysis are created by the function:

ProjByComp: contains the annotations of the features or genes, one file per component;

varAnalysisOnA: contains two directories: 'qual/' and 'quant/' which respectively contain the results of the association between components qualitative and quantitative variables;

Heatmaps: contains the heatmaps (one pdf file per component) of contributing genes by component;

varOnSampleHist: contains athe histograms of sample contributions superimposed with the histograms of the samples grouped by variable;

cluster2var: contains the association between a clustering of the samples performed on the mixing matrix `A` and the variables.

Value

NULL

Author(s)

Anne Biton

See Also

`writeProjByComp`,

Examples

```
## Not run:

## load an example of IcaSet
data(icaSetCarbayo)
## make sure the 'mart' attribute is correctly defined
mart(icaSetCarbayo) <- useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")

## creation of an object of class MineICAParams
## here we use a low threshold because 'icaSetCarbayo' is already
# restricted to the contributing features/genes
```

```
params <- buildMineICAParams(resPath="~/resMineICACarbayotestRunAn/", selCutoff=2, pvalCutoff=0.05)
require(hgu133a.db)

runAn(params=params, icaSet=icaSetCarbayo)

## End(Not run)
```

---

runCompareIcaSets *runCompareIcaSets*

Description

This function encompasses the comparison of several IcaSet objects using correlations and the plot of the corresponding correlation graph. The IcaSet objects are compared by calculating the correlation between either projection values of common features or genes, or contributions of common samples.

Usage

```
runCompareIcaSets(icaSets, labAn,
  type.corr = c("pearson", "spearman"), cutoff_zval = 0,
  level = c("genes", "features", "samples"),
  fileNodeDescr = NULL, fileDataGraph = NULL,
  plot = TRUE, title = "", col, cutoff_graph = NULL,
  useMax = TRUE, tkplot = FALSE)
```

Arguments

icaSets	List of IcaSet objects, e.g results of ICA decompositions obtained on several datasets.
labAn	Vector of names for each icaSet, e.g the the names of the datasets on which were calculated the decompositions.
type.corr	Type of correlation to compute, either 'pearson' or 'spearman'.
cutoff_zval	Either NULL or 0 (default) if all genes are used to compute the correlation between the components, or a threshold to compute the correlation using the genes that have at least a scaled projection higher than cutoff_zval. Will be used only when level is one of c("features","genes").
level	Data level of the IcaSet objects on which is applied the correlation. It must correspond to a data level shared by the IcaSet objects: 'samples' if they were applied to common samples (correlations are computed between matrix A), 'features' if they were applied to common features (correlations are computed between matrix S), 'genes' if they share gene IDs after annotation into genes (correlations are computed between matrix SByGene).
fileNodeDescr	File where node descriptions are saved (useful when the user wants to visualize the graph using Cytoscape).
fileDataGraph	File where graph description is saved (useful when the user wants to visualize the graph using Cytoscape).
plot	if TRUE (default) plot the correlation graph
title	title of the graph

col	vector of colors indexed by elements of labAn; if missing, colors will be automatically attributed
cutoff_graph	the cutoff used to select pairs that will be included in the graph
useMax	if TRUE, the graph is restricted to edges that correspond to maximum correlation between components, see details
tkplot	If TRUE, performs interactive plot with function tkplot, else uses plot.igraph

Details

This function calls four functions: compareAn which computes the correlations, compareAn2graphfile which builds the graph, nodeAttrs which builds the node description data, and plotCorGraph which uses tkplot to plot the graph in an interactive device.

If the user wants to see the correlation graph in Cytoscape, he must fill the arguments fileDataGraph and fileNodeDescr, in order to import the graph and its node descriptions as a .txt file in Cytoscape.

When labAn is missing, each element i of icaSets is labeled as 'Ani'.

The user must carefully choose the data level used in the comparison: If level='samples', the correlations are based on the mixing matrices of the ICA decompositions (of dimension samples x components). 'A' will be typically chosen when the ICA decompositions were computed on the same dataset, or on datasets that include the same samples. If level='features' is chosen, the correlation is calculated between the source matrices (of dimension features x components) of the ICA decompositions. 'S' will be typically used when the ICA decompositions share common features (e.g same microarrays). If level='genes', the correlations are calculated on the attributes 'SByGene' which store the projections of the annotated features. 'SByGene' will be typically chosen when ICA were computed on datasets from different technologies, for which comparison is possible only after annotation into a common ID, like genes.

cutoff_zval is only used when level is one of c('features','genes'), in order to restrict the correlation to the contributing features or genes.

When cutoff_zval is specified, for each pair of components, genes or features that are included in the circle of center 0 and radius cutoff_zval are excluded from the computation of the correlation.

It must be taken into account by the user that if cutoff_zval is different from NULL or zero, the computation will be much slowler since each pair of component is treated individually.

Edges of the graph are built based on the correlation values between the components. Absolute values of correlations are used since components have no direction.

If useMax is TRUE each component will be linked to only one component of each other IcaSet that corresponds to the most correlated component among all components of the same IcaSet. If cutoff_graph is specified, only correlations exceeding this value are taken into account to build the graph. For example, if cutoff is 1, only relationships between components that correspond to a correlation value higher than 1 will be included. Absolute correlation values are used since the components have no direction.

The contents of the returned list are

dataGraph: dataGraph data.frame that describes the correlation graph,

nodeAttrs: nodeAttrs data.frame that describes the node of the graph

graph graph the graph as an igraph-object,

graphid: graphid the id of the graph plotted using tkplot.

Value

A list consisting of

dataGraph: a data.frame defining the correlation graph

nodeAttrs: a data.frame describing the node of the graph,

graph: the graph as an object of class `igraph`,

graphid the id of the graph plotted with `tkplot`.

Author(s)

Anne Biton

See Also

`compareAn2graphfile`, `compareAn`, `cor2An`, `plotCorGraph`

Examples

```
dat1 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat1) <- paste("g", 1:1000, sep="")
colnames(dat1) <- paste("s", 1:10, sep="")
dat2 <- data.frame(matrix(rnorm(10000),ncol=10,nrow=1000))
rownames(dat2) <- paste("g", 1:1000, sep="")
colnames(dat2) <- paste("s", 1:10, sep="")

## run ICA
resJade1 <- runICA(X=dat1, nbComp=3, method = "JADE")
resJade2 <- runICA(X=dat2, nbComp=3, method = "JADE")

## build params
params <- buildMineICAParams(resPath="toy/")

## build IcaSet objects
icaSettoy1 <- buildIcaSet(params=params, A=data.frame(resJade1$A), S=data.frame(resJade1$S),
                          dat=dat1, alreadyAnnot=TRUE)$icaSet
icaSettoy2 <- buildIcaSet(params=params, A=data.frame(resJade2$A), S=data.frame(resJade2$S),
                          dat=dat2, alreadyAnnot=TRUE)$icaSet

## compare IcaSet objects
## use tkplot=TRUE to get an interactive graph
rescomp <- runCompareIcaSets(icaSets=list(icaSettoy1, icaSettoy2), labAn=c("toy1","toy2"),
                             type.corr="pearson", level="genes", tkplot=FALSE)


## Not run:
## load the microarray-based gene expression datasets
## of breast tumors
library(breastCancerMAINZ)
library(breastCancerVDX)
data(mainz)
data(vdx)

## Define a function used to build two examples of IcaSet objects
## and annotate the probe sets into gene Symbols
treat <- function(es, annot="hgu133a.db") {
```

```
   es <- selectFeatures_IQR(es,10000)
   exprs(es) <- t(apply(exprs(es),1,scale,scale=FALSE))
   colnames(exprs(es)) <- sampleNames(es)
   resJade <- runICA(X=exprs(es), nbComp=10, method = "JADE", maxit=10000)
  resBuild <- buildIcaSet(params=buildMineICAParams(), A=data.frame(resJade$A), S=data.frame(resJade$S),
                      dat=exprs(es), pData=pData(es), refSamples=character(0),
                      annotation=annot, typeID= typeIDmainz,
                      chipManu = "affymetrix", mart=mart)
   icaSet <- resBuild$icaSet
}
## Build the two IcaSet objects
icaSetMainz <- treat(mainz)
icaSetVdx <- treat(vdx)

## compare the IcaSets
runCompareIcaSets(icaSets=list(icaSetMainz, icaSetVdx), labAn=c("Mainz","Vdx"), type.corr="pearson", level=

## End(Not run)
```

`runEnrich` *Enrichment analysis through GOstats*

Description

This function tests the enrichment of the components of an `IcaSet` object using package `GOstats` through function `hyperGTest`.

Usage

```
runEnrich(icaSet, params, dbs = c("KEGG", "GO"),
  ontos = c("BP", "CC", "MF"), cond = TRUE,
  hgCutoff = params["pvalCutoff"])
```

Arguments

`icaSet`	An object of class `IcaSet`
`params`	An object of class `MineICAParams` providing the parameters of the analysis
`dbs`	The database to use, default is `c("GO","KEGG")`
`ontos`	A string specifying the GO ontology to use. Must be one of `"BP"`, `"CC"`, or `"MF"`, see `GOHyperGParams-class`. Only used when argument `dbs` includes `"GO"`.
`cond`	A logical indicating whether the calculation should condition on the GO structure, see `GOHyperGParams-class`. Only used when argument `dbs` includes `"GO"`.
`hgCutoff`	The threshold p-value for statistical significance, default is `pvalCutoff(params)`

Details

An annotation package should be available in `annotation(icaSet)` to provide the contents of the gene sets. If none corresponds to the technology you deal with, please choose the org.*.eg.db package according to the organism (for example org.Hs.eg.db for Homo sapiens). By default, if `annotation(icaSet)` is empty and organism is one of `c("Human","HomoSapiens","Mouse","Mus Musculus")`, then either `org.Hs.eg.db` or `org.Mm.eg.db` is used.

Use of GOstats requires the input IDs to be Entrez Gene, this function will therefore annotate either the feature names or the gene names into Entrez Gene ID using either the annotation package (annotation(icaSet)) or biomaRt.

Three types of enrichment tests are computed for each component: the threshold is first used to select gene based on their absolute projections, then positive and negative projections are treated individually.

For each database db (each ontology if db is "GO"), this function writes an HTML file containing the outputs of the enrichment tests computed through the function hyperGTest. The corresponding files are located in resPath(icaSet)/GOstatsEnrichAnalysis/byDb/. The results obtained for each database/ontology are then merged into an array for each component, this array is written as an HTML file in the directory resPath(icaSet)/GOstatsEnrichmentAnalysis/ (this directory is first deleted if it already exists). This file is the one the user should look at.

The outputs of hyperGTest that are given in each table are:

DB, ID, Term: the database, the gene set ID, and the gene Set name

P-value: probability of observing the number of genes annotated for the gene set among the selected gene list, knowing the total number of annotated genes among the universe,

Expected counts: expected number of genes in the selected gene list to be found at each tested category term/gene set,

Odds ratio: odds ratio for each category term tested which is an indicator of the level of enrichment of genes within the list as against the universe,

Counts: number of genes in the selected gene list that are annotated for the gene set,

Size: number of genes from the universe annotated for the gene set.

Value

NULL

Author(s)

Anne Biton

See Also

buildIcaSet, useMart, hyperGTest, GOHyperGParams, hypergeoAn, mergeGostatsResults

Examples

```
## Not run:
# Load examples of IcaSet object
data(icaSetCarbayo)

## Define parameters
# Use threshold 3 to select contributing genes on which enrichment analysis will be applied
# Results of enrichment analysis will be written in path 'resPath(params)/GOstatsEnrichAnalysis'
params <- buildMineICAParams(resPath="carbayo/", selCutoff=3)

## Run enrichment analysis on the first two components contained in the icaSet object 'icaSetCarbayo'
runEnrich(params=params,icaSet=icaSetCarbayo[,,1:2],dbs="GO", ontos="BP")


## End(Not run)
```

runICA *Run of fastICA and JADE algorithms*

Description

This function performs ICA decomposition of a matrix using functions fastICA and JADE.

Usage

```
runICA(method = c("fastICA", "JADE"), X, nbComp,
  alg.type = c("deflation", "parallel"),
  fun = c("logcosh", "exp"), maxit = 500, tol = 10^-6,
  ...)
```

Arguments

method	The ICA method to use, either "JADE" (the default) or "fastICA".
X	A data matrix with n rows representing observations (e.g genes) and p columns representing variables (e.g samples).
nbComp	The number of components to be extracted.
alg.type	If alg.type="parallel" the components are extracted simultaneously (the default), if alg.type="deflation" the components are extracted one at a time, see fastICA.
fun	The functional form of the G function used in the approximation to neg-entropy (see 'details' of the help of function fastICA).
maxit	The maximum number of iterations to perform.
tol	A positive scalar giving the tolerance at which the un-mixing matrix is considered to have converged.
...	Additional parameters for fastICA and JADE

Details

See details of the functions fastICA and JADE.

Value

A list, see outputs of fastICA and JADE. This list includes at least three elements:

A the estimated mixing matrix

S the estimated source matrix, itemWthe estimated unmixing matrix

Author(s)

Anne Biton

Examples

```
set.seed(2004);
M <- matrix(rnorm(5000*6,sd=0.3),ncol=10)
M[1:10,1:3] <- M[1:10,1:3] + 2
M[1:100,1:3] <- M[1:100,1:3] +1
resJade <- runICA(X=M, nbComp=2, method = "JADE", maxit=10000)
```

`selectContrib` *Select contributing features/genes*

Description

This function selects elements whose absolute scaled values exceed a given threshold.

Usage

```
selectContrib(object, cutoff, level, ...)
```

Arguments

`object`	Either an `IcaSet` object, or a list of projection vectors, e.g the list of feature or gene projections on each component.
`cutoff`	The threshold according to which the elements will be selected. Must be either of length 1 and the same treshold is applied to all components, or of length equal to the number of components in order to use a specific threshold for each component.
`level`	The level of the selection: either `"genes"` to select contributing genes using `SByGene(icaSet)`, or `"features"` to select contributing features using `S(icaSet)`.
`...`	...

Details

Each vector is first scaled and then only elements with an absolute scaled value higher than `cutoff` are kept.

Value

A list of projections restricted to the elements that are higher than `cutoff`.

Author(s)

Anne Biton

Examples

```
## Not run:
## load an example of icaSet
data(icaSetCarbayo)

##### =========
####  When arg 'object' is an IcaSet object
##### =========

## select contributing genes
selectContrib(object=icaSetCarbayo, cutoff=3, level="genes")

## select contributing features
selectContrib(object=icaSetCarbayo, cutoff=3, level="features")
```

```
##### =========
#### When arg 'object' is a list
##### =========
 c1 <- rnorm(100); names(c1) <- 100:199
 c2 <- rnorm(100); names(c2) <- 1:99
 selectContrib(object=list(c1,c2), cutoff= 0.5)

## select contributing features
contribFlist <- selectContrib(Slist(icaSetCarbayo), 3)

## select contributing genes
contribGlist <- selectContrib(SlistByGene(icaSetCarbayo), 3)



## End(Not run)
```

`selectFeatures_IQR` *Selection of features based on their IQR*

Description

This function selects the features having the largest Inter Quartile Range (IQR).

Usage

```
selectFeatures_IQR(data, nb)
```

Arguments

`data`	Measured data of dimension features x samples (e.g, gene expression data)
`nb`	The number of features to be selected

Value

A subset of `data` restricted to the features having the `nb` highest IQR value

Author(s)

Pierre Gestraud

Examples

```
dat <- matrix(rnorm(10000),ncol=10,nrow=1000)
rownames(dat) <- 1:1000
selectFeatures_IQR(data=dat, nb=500)
```

selectWitnessGenes *selectWitnessGenes*

Description

This function selects a gene per component.

Usage

```
selectWitnessGenes(icaSet, params,
  level = c("genes", "features"), maxNbOcc = 1,
  selectionByComp = NULL)
```

Arguments

icaSet	An object of class IcaSet
params	An object of class MineICAParams containing the parameters of the analysis, the attribute cutoffSel is used as the threshold.
level	The attribute of icaSet to be used, the witness elements will be either selected within the "features" or the "genes"
maxNbOcc	The maximum number of components where the genes can have an absolute projection value higher than cutoffSel(params) in order to be selected.
selectionByComp	The list of components already restricted to the contributing genes

Details

Selects as feature/gene witness, for each component, the first gene whose absolute projection is greater than a given threshold in at the most maxNbOcc components. These witnesses can then be used as representatives of the expression behavior of the contributing genes of the components.

When a feature/gene respecting the given constraints is not found, maxNbOcc is incremented of one until a gene is found.

Value

This function returns a vector of IDs.

Author(s)

Anne Biton

Examples

```
## load an example of IcaSet
data(icaSetCarbayo)

## define parameters: features or genes are considered to be contributor
# when their absolute projection value exceeds a threshold of 4.
params <- buildMineICAParams(resPath="carbayo/", selCutoff=4)

## selection, as gene witnesses, of the genes whose absolute projection is greater than 4
```

```
# in at the most one component. I.e, a gene is selected as a gene witness of a component
# if he has a large projection on this component only.
selectWitnessGenes(icaSet=icaSetCarbayo, params=params, level="genes", maxNbOcc=1)

## selection, as gene witnesses, of the genes whose absolute projection is greater than 4
# in at the most two components.
# I.e, a gene is selected as a gene witness of a given component if he has a large projection
# in this component and at the most another.
selectWitnessGenes(icaSet=icaSetCarbayo, params=params, level="genes", maxNbOcc=2)
```

`Slist` *Retrieve feature/gene projections stored in an* `IcaSet` *object as a list.*

Description

These generic functions retrieve, from an IcaSet object, the feature and gene projections contained in the attribute `S` and `SByGene` as a list where feature and gene IDs are preserved.

Usage

```
Slist(object)
SlistByGene(object)
```

Arguments

`object`	Object of class `IcaSet`.

Value

`Slist` and `SlistByGene` return a list whose length equals the number of components contained in the `IcaSet` object. Each element of this list contains a vector of feature or gene projections indexed by the feature or gene IDs.

Author(s)

Anne Biton

See Also

`IcaSet-class`

writeGenes *Description of features using package* biomaRt.

Description

This function annotates IDs (typically gene IDs) provided by the user and returns an html file with their description.

Usage

```
writeGenes(data, filename = NULL,
  mart = useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl"),
  typeId = "hgnc_symbol", typeRetrieved = NULL,
  sortBy = NULL, sortAbs = TRUE, colAnnot = NULL,
  decreasing = TRUE, highlight = NULL, caption = "")
```

Arguments

data	Either a data.frame whose rownames or one of its columns contain the IDs to be annotated, or a vector of IDs.
filename	The name of the HTML file where gene annotations are written.
mart	Output of function useMart from package biomaRt.
typeId	The type of IDs available in data, in the biomaRt way (type listFilters(mart) to choose one).
typeRetrieved	The descriptors uses to annotate the features of data (type listAttributes(mart) to choose one or several).
sortBy	Name of a column of data used to order the output.
sortAbs	If TRUE absolute value of column sortBy is used to order the output.
colAnnot	The column containing the IDs to be annotated, if NULL or missing and argument data is a data.frame, then rownames of data must contain the IDs.
decreasing	If TRUE, the output is sorted by decreasing values of the sortBy column
highlight	IDs to be displayed in colour red in the returned table
caption	A title for the HTML table

Details

"hgnc_symbol", "ensembl_gene_id", "description", "chromosome_name", "start_position", "end_posit and "strand", are automatically added to the list of fields available in argument typeRetrieved queried on biomaRt. The web-links to www.genecards.org and www.proteinatlas.org are automatically added in the columns of the output respectively corresponding to hgnc_symbol and ensembl_gene_id.

Value

This function returns a data.frame which contains annotations of the input data.

Author(s)

Anne Biton

See Also

getBM, listFilters, listAttributes, useMart

Examples

```
if (interactive()) {
## define the database to be used
mart <- useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")

### Describe:
## a set of hgnc symbols with default descriptions (typeRetrieved=NULL)
genes <- c("TOP2A","E2F3","E2F1","CDK1","CDC20","MKI67")
writeGenes(data=genes, filename="foo", mart=mart, typeId = "hgnc_symbol")

## a data.frame indexed by hngc symbols, sort output according to column "values", add a title to the HTML outp
datagenes <- data.frame(values=rnorm(6),row.names = genes)
writeGenes(data=datagenes, filename="foo", sortBy = "values", caption = "Description of some proliferation ge

## a set of Entrez Gene IDs with default descriptions
genes <- c("7153","1871","1869","983","991","4288")
writeGenes(data=genes, filename="foo", mart=mart, typeId = "entrezgene")
}
## Not run:
## add the GO category the genes belong to
## search in listAttributes(mart)[,1] which filter correspond to the Gene Ontology -> "go_id"
writeGenes(data=genes, filename="foo", mart=mart, typeId = "entrezgene", typeRetrieved = "go_id")

## End(Not run)
```

---

writeProjByComp *writeProjByComp*

Description

This function writes in an html file the description of the features, or genes, that contribute to each component. It also writes an html file containing, for each feature or gene, its projection value on every component.

Usage

```
writeProjByComp(icaSet, params, mart = useMart(biomart = "ensembl",
    dataset = "hsapiens_gene_ensembl"), typeRetrieved = NULL, addNbOcc =
   TRUE, selectionByComp = NULL, level = c("features", "genes"), typeId, selCutoffWrite=2.5)
```

Arguments

icaSet	An object of class IcaSet
params	An object of class MineICAParams containing the parameters of the analysis. The files are written in the path genesPath(params). selCutoff(params) is used to select the features or genes by component.
mart	An output of function useMart containing the database used for annotation.

typeRetrieved	The annotations biomaRt is queried about. They describe the feature or gene IDs of the argument icaSet, see listFilters.
addNbOcc	If TRUE, the number of components the features/genes contribute to is added to the output. A gene/feature is considered as a contributor of a component if its absolute scaled projection value is higher than selCutoff(icaSet).
selectionByComp	A list containing the feature/gene projections on each component, already restricted to the ones considered as contributors.
level	The data level of icaSet that will be annotated: either the feature projections ("features"), or the gene projections ("genes").
typeId	The type of ID the features or the genes of icaSet correspond to. By default typeID(icaSet) is used. It must be provided in the biomaRt way (type listFilters(mart) to choose the appropriate value).
selCutoffWrite	The cutoff applied to the absolute projection values to select the features/genes that will be annotated using package biomaRt, default is 2.5.

Details

One file is created by component, each file is named by the index of the components (indComp(icaSet)) and located in the path genePath(params).

In case you are interested in writing the description of features and their annotations, please remember to modify codegenesPath(params), or the previous files will be overwritten.

The genes are ranked according to their absolute projection values.

This function also writes an html file named "genes2comp" providing, for each feature or gene, the number of components it contributes to (according to the threshold cutoffSel(params)), and its projection value on all the components. The projection values are scaled.

See function writeGenes for details.

Value

This function returns a list of two elements:

listAnnotComp: a list with the output of writeGenes for each component

nbOccInComp: a data.frame storing the projection values of each feature/gene (row) across all the components (columns).

Author(s)

Anne Biton

See Also

writeGenes, getBM, listFilters, listAttributes, useMart, selectContrib, nbOccInComp

Examples

```
## Not run:
## load IcaSet object
## We will use 'icaSetCarbayo', whose features are hgu133a probe sets
## and feature annotations are Gene Symbols.
data(icaSetCarbayo)
```

```
## define database to be used by biomaRt
mart <- useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")

## define the parameters of the analysis
params <- buildMineICAParams(resPath="~/resMineICACarbayo/", selCutoff=0)

## Make sure the elements "_biomaRt" of attribute 'typeID' are defined
typeID(icaSetCarbayo)

### Query biomaRt and write gene descriptions in HTML files
### The files will be located in the directory 'genesPath(params)'

## 1. Write description of genes
res <- writeProjByComp(icaSet=icaSetCarbayo, params=params, mart=mart,
           level="genes") #, typeId="hgnc_symbol")

## 2. Write description of features
# change attribute 'genesPath' of params to preserve the gene descriptions
genesPath(params) <- paste(resPath(params),"comp2features/",sep="")
res <- writeProjByComp(icaSet=icaSetCarbayo, params=params, mart=mart,
           level="features") #, typeId="affy_hg_u133a")


## End(Not run)
```

`writeRnkFiles` *Write rnk files containing gene projections*

Description

Writes the gene projection values of each component in a '.rnk' file for GSEA.

Usage

```
writeRnkFiles(icaSet, abs = TRUE, path)
```

Arguments

`icaSet`	An object of class `IcaSet`
`abs`	If TRUE (default) the absolute projection values are used.
`path`	The path that will contain the rnk files.

Details

The .rnk format requires two columns, the first containing the gene IDs, the second containing the projection values. The genes are ordered by projection values. The files are named "index-of-component_abs.rnk" if `abs=TRUE`, or "index-of-component.rnk" if `abs=FALSE`.

Value

NULL

Author(s)

Anne

Index