Package ‘mogsa’

April 14, 2017

Type Package
Title Multiple omics data integrative clustering and gene set analysis
Version 1.8.0
Date 2016-04-05
Author Chen Meng
Maintainer Chen Meng <mengchen18@gmail.com>
Description This package provide a method for doing gene set analysis based on multiple omics data.
License GPL-2
Depends R (>= 3.2.0)
Imports methods, graphite, genefilter, BiocGenerics, gplots, GSEABase,
   Biobase, parallel, corpcor, svd, cluster
VignetteBuilder knitr
Suggests BiocStyle, knitr
biocViews GeneExpression, PrincipalComponent, StatisticalMethod,
   Clustering, Software
NeedsCompilation no

R topics documented:

mogsa-package .......................................................... 2
annotate.gs .......................................................... 3
bootMbpca .......................................................... 4
bootMbpcaK ......................................................... 5
box.gs.feature ..................................................... 6
combine-methods .................................................. 7
decompose.gs.group .............................................. 8
decompose.gs.ind ................................................ 9
deflat ............................................................. 10
distMoa ............................................................ 11
getmgsa ............................................................ 12
GIS ................................................................. 13
matpower .......................................................... 14
mbpca .............................................................. 15
mgsa-class ......................................................... 17
The main function in the package is "mogsa", see the function help manu for more details.
Author(s)

Chen Meng
Maintainer: Chen Meng <chen.meng@tum.de>

References

Chen Meng, Dominic Helm, Martin Frejno, and Bernhard Kuster. moCluster: Identifying Joint Patterns Across Multiple Omics Data Sets. Journal of Proteome Research 2016.

Examples

# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)

# using a list of data.frame as input
mgsa1 <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)

# using moa as input
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
smoa <- sup.moa(ana, sup=NCI60_4array_supdata, nf=3)
mgsa2 <- mogsa(x = ana, sup=NCI60_4array_supdata, nf=3)
mgsa3 <- mogsa(x = ana, sup=smoa)

#### annotate.gs

**Summary annotation information of a gene set**

**Description**

Retrieve variables/features (genes) mapped to the annotated data sets in a gene set. Also returns the information about presence and absence of a feature for a specific data set.

**Usage**

annotate.gs(mgsa, gs)

**Arguments**

- mgsa: An object of class mogsa-class or moa.sup-class.
- gs: The name of a geneset

**Value**

This function returns a data.frame. The first column shows the name of features. The last column is for the count of how many data sets has the corresponding features. Columns in the middle contains logical value indicating whether a feature is presented in a particular data set.

Author(s)

Chen Meng
See Also

see GIS

Examples

```r
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
msa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
              proc.row = "center_ssq!", w.data = "inertia", statis = TRUE)
allgs <- colnames(NCI60_4array_supdata[[1]])
annotate.gs(msa, allgs[1])
```

---

**bootMbpca**

*Bootstrap mbpca to estimate the coherence of different data sets*

Description

Bootstrap mbpca to estimate the coherence of different data sets and estimate the number of components should be included in an analysis.

Usage

```r
bootMbpca(moa, mc.cores = 1, B = 100, replace = TRUE,
          resample = c("sample", "gene", "total"), log = "y", ncomp = NULL, method = NULL,
          maxiter = 1000, svd.solver = c("svd", "fast.svd", "propack"), plot = TRUE)
```

Arguments

- **moa**
  An object of `moa` returned by `mbpca`.

- **mc.cores**
  Integer; number of cores used in bootstrap. This value is passed to function `mclapply`.

- **B**
  Integer; number of bootstrap.

- **replace**
  Logical; sampling with or without replacement.

- **resample**
  Could be one of "sample", "gene" or "total". "sample" and "gene" means sample-wise and variable-wise resampling, respectively. "total" means total resampling.

- **log**
  Could be "x", "y" or "xy" for plot log axis.

- **ncomp**
  Passed to function `mbpca`. In most of cases, user do not need to specify this argument because it could be inferred from `moa`.

- **method**
  Passed to function `mbpca`. In most of cases, user do not need to specify this argument because it could be inferred from `moa`.

- **maxiter**
  Passed to function `mbpca`. In most of cases, user do not need to specify this argument because it could be inferred from `moa`.

- **svd.solver**
  Passed to function `mbpca`. In most of cases, user do not need to specify this argument because it could be inferred from `moa`.

- **plot**
  Logical; whether the result should be plotted.
Details
update details.

Value
It returns a matrix, columns are eigenvalues for different components. Each rows is a bootstrap sample.

Author(s)
Chen Meng

Examples
# see examples in \code{\link{mbpca}}

---

bootMbpcaK  
*An internal function called by* bootMbpca.

Description
An internal function called by bootMbpca.

Usage

```r
bootMbpcaK(data, replace, B = 100, mc.cores = 1, resample = c("sample", "total", "gene"),
ncomp, method, k, center = FALSE, scale = FALSE, option = "uniform", maxiter = 1000,
svd.solver = c("svd", "fast.svd", "propack"))
```

Arguments

- `data`: A list of matrix to bootstrap.
- `replace`: A logical variable to indicate sampling with or without replacement.
- `B`: Integer; number of bootstrap.
- `mc.cores`: Integer; number of cores used in bootstrap. This value is passed to function mclapply.
- `resample`: Could be one of "sample", "gene" or "total". "sample" and "gene" means sample-wise and variable-wise resampling, respectively. "total" means total resampling.
- `ncomp`: passed to mbpca.
- `method`: passed to mbpca.
- `k`: passed to mbpca.
- `center`: passed to mbpca.
- `scale`: passed to mbpca.
- `option`: passed to mbpca.
- `maxiter`: passed to mbpca.
- `svd.solver`: passed to mbpca.
Value
A matrix of mbpca eigenvalues resulted from bootstrap samples

Author(s)
Chen Meng

See Also
bootMbpca

box.gs.feature

Description
boxplot of gene set variables across all samples.

Usage
box.gs.feature(x, gs, moa = NULL, col = 1, layout = NULL, plot = TRUE, obs.order = NULL, ...)

Arguments
x An object of class mgsa-class or moa.sup-class
gs Gene set want to be explored
moa An object of class moa. It is required if x is an object of class moa.sup-class
col The color code for samples
layout The layout control, see examples.
plot A logical indicates whether the result should be plotted. If FALSE, a list of expression matrix of the gene set genes is returned. Otherwise nothing returned.
obs.order Can be used to reorder the matrix, could be used when clustering result is available.
... The arguments passed to boxplot

Details
This is a convenient function used to explore the expression of a set of features/genes

Value
Do not return anything (plot=TRUE) or return a list of matrix (plot=FALSE) depends on plot argument.

Author(s)
Chen meng
Examples

# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
mgsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
    proc.row = "center_ssq!", w.data = "inertia", statis = TRUE)

allgs <- colnames(NCI60_4array_supdata[[1]])
colcode <- as.factor(sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\."), ",[", 1))
a <- box.gs.feature(x=mgsa, gs=allgs[5], type=3, col=colcode, plot=FALSE)
box.gs.feature(x=mgsa, gs=allgs[5], type=3, col=colcode, plot=TRUE, layout=matrix(1:4, 2, 2))

combine-methods

Combine two objects of class mgsa into one.

Description

This function could only be used to combine two "mgsa" objects at present; using "Reduce" function to combine more.

Usage

combine(x, y, ...)

Arguments

x one mgsa object
y another mgsa object
... ignored. Only two mgsa objects could be combined, using "Reduce" to combine more than two sets.

Value

A combined object of class mgsa will be returned.

Methods

signature(x = "mgsa", y = "mgsa") To combine two objects of mgsa.

This function could only be used to combine two "mgsa" objects; using "Reduce" function to combine more.

Examples

# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
# split gene set annotation into two sets.
sup1 <- lapply(NCI60_4array_supdata, function(x) x[, 1:10])
sup2 <- lapply(NCI60_4array_supdata, function(x) x[, -(1:10)])
# project two sets of annotation
decompose.gs.group <- mogsa(x = NCI60_4arrays, sup=sup1, nf=9,
                         proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
mgsa2 <- mogsa(x = NCI60_4arrays, sup=sup2, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
# combine two independent mgsa sets
mgsa_comb <- combine(mgsa1, mgsa2)
dim(getmgsa(mgsa1, "score"))
dim(getmgsa(mgsa2, "score"))
dim(getmgsa(mgsa_comb, "score"))

---

decompose.gs.group  Data-wise or PC-wise decomposition of gene set scores for all observations.

**Description**

Data-wise or PC-wise decomposition of gene set scores (GSS) across all observations. The pre-defined group/cluster information should be given so that the mean decomposed GSSs for each group are returned and plotted.

**Usage**

decompose.gs.group(x, gs, group, decomp = "data", nf = 2, x.legend = "bottomleft",
y.legend = NULL, plot = TRUE, main = NULL, ...)

**Arguments**

- **x**: An object of class `mgsa-class` or `moa.sup-class`
- **gs**: The gene set want to exam.
- **group**: An vector or factor to indicate the group of observations, such as clusters. See examples.
- **decomp**: A character string either "data" or "pc" to indicate how the gene set scores should be decomposed (with respect to data or PC).
- **nf**: The number of axes/PCs to be calculated and plotted.
- **x.legend**: Used to control the position of legends.
- **y.legend**: Used to control the position of legends.
- **plot**: A logical indicates if a plot should be drawn.
- **main**: The main title of plot.
- **...**: Other arguments passed to `barplot`.

**Details**

This function could be used when the number of observation is large and there are cluster/group information is available. In this case, the means of decomposed gene set scores over each group is calculated. The vertical bar on the end of each bar indicates the 95% confident interval of the means.

**Value**

Return nothing or a matrix depends on how argument `plot` is set.
**decompose.gs.ind**

**Author(s)**
Chen Meng

**References**
TBA

**See Also**
See Also `decompose.gs.ind`

**Examples**

```r
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)

# using a list of data.frame as input
mgsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
              proc.row = "center_ssq!", w.data = "inertia", statis = TRUE)

colcode <- as.factor(sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\."), 
                           
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
Details

type=1 (the data-pc mode), the axes/PCs are represented as the narrow bars with different colors and the background wide bars behind narrow bars are gene set scores for datasets, which is calculated from the sum of all underlying individual axes/PC scores. When type=2 (the pc-data mode) the interpretation of narrow and wide bars are in the other way around. If type=3, both are shown.

This function could only be used to check the decomposition of gene set scores of a single observation. So the function is not efficient when the number of observation is large. Another function `decompose.gs.group`, could be used in this case, particularly when the cluster information of the observation panel is available.

Value

Return nothing or a matrix depends on how argument `plot` is set.

Author(s)

Chen Meng

References

TBA

See Also

See Also as `decompose.gs.group`

Examples

```r
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
mgsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
proc.row = "center_ssq!", w.data = "inertia", statis = TRUE)
allgs <- colnames(NCI60_4array_supdata[[1]])
# plot
decompose.gs.ind(x=mgsa, gs=allgs[5], obs="BR.MDA_MB_231", type=2, nf=5)
# or
decompose.gs.ind(x=getmgsa(mgsa, "sup"), gs=allgs[5], obs="BR.MDA_MB_231", type=3, nf=5)
```

`deflat`

`deflat` function used by `mbpca`

Description

An internal function called by `mbpca`.

Usage

`deflat(x, t, tb, pb, method = "globalScore")`
distMoa

Arguments

x A list of matrix want to deflat

The global scores returned by msvd or nipalsSoftK

tb The block scores returned by msvd or nipalsSoftK

The block loadings returned by msvd or nipalsSoftK

method A character to specify the deflation strateg, could be one of c("globalScore", "blockLoading", "blockScore").

Value

A list of deflated matrix

Author(s)

Chen Meng

----------

distMoa

*Calculate the distance matrix from an object of class moa-class.*

Description

A convenient function to calculate the distance matrix from an object of class moa-class.

Usage

```
distMoa(x, nf = NA, tol = 1e-05, method = "euclidean", 
        diag = FALSE, upper = FALSE, p = 2)
```

Arguments

x An object of class moa-class.

nf Integer; the number of component used to calculate the distance. Default setting (NA) will keep all the axes.

tol Numerical; the tolerance of component with low variance.

method passed to function dist

diag passed to function dist

upper passed to function dist

p passed to function dist

Value

An object of class dist, see function "dist".

Author(s)

Chen Meng
getmgsa

Examples

```r
# see examples in \code{\link{mbpca}}

data("NCI60_4arrays")
moa <- mbpca(NCI60_4arrays, ncomp = 10, k = "all", method = "globalScore", option = "lambda1",
            center=TRUE, scale=FALSE)

dst <- distMoa(moa)
```

getmgsa

get values in an object of class "mgsa".

Description

get values/slot in an object of class "mgsa". The "mgsa" consists of two S4 class objects, \code{moa-class} and \code{moa.sup-class}. This function could extract values in these two components directly.

Usage

```r
getmgsa(mgsa, value)
```

Arguments

- `mgsa`: An object of class \code{mgsa-class}.
- `value`: The name of the value want to extract from "mgsa". See detail for options.

Details

if value in c("call", "moa", "sup"), the function equal function \code{slot}.
if value in c("eig", "tau", "partial.eig", "eig.vec", "loading", "fac.scr", "partial.fs", "ctr.obs", "ctr.var",
            "ctr.tab", "RV"), the function extract corresponding value from \code{moa-class}.
if value in c("data", "coord.sep", "coord.comb", "score", "score.data", "score.pc", "score.sep", "p.val"),
the function extract value from \code{moa.sup-class}.

Value

The function return the selected value in "mgsa".

Author(s)

Chen Meng

References

TBA
Examples

```r
# library(mogsa)
data(NCI60_4array_supdata)
data(NCI60_4arrays)
mgsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
part.eig <- getmgsa(mgsa, "partial.eig")
barplot(as.matrix(part.eig))
```

GIS

**calculate gene influential scores of genes in a gene set.**

Description

Calculate the gene influential score of individual feature to the overall variance of GS score. Using a leave-one-out procedure (See detail).

Usage

```r
GIS(x, geneSet, nf=NA, barcol=NA, topN=NA, plot=TRUE, Fvalue=FALSE, ff=NA, cor=FALSE)
```

Arguments

- **x** An object of class `mgsa-class`.
- **geneSet** A character string or number to indicated the gene sets under consideration.
- **nf** The number of PCs used in the calculation of gene set scores. The default is NA, which means using all the PCs in the mogsa. This should work for most of the cases.
- **barcol** The color of the bars, which is used to distinguish features/genes from different datasets, so its length should be the same as the number of data sets.
- **topN** An positive integer specify the number of top influencers that should to returned.
- **plot** A logical indicate if the result should be plotted.
- **Fvalue** A logical indicate if the GIS should be calculated in a supervised manner.
- **ff** The vector indicates the group of columns for calculating the F-ratio when Fvalue=TRUE.
- **cor** A logical indicates whether use correlation between reconstructed expression with GSS. This is faster than the standard GIS.

Details

The evaluation of the importance of a single feature is calculated in the supervised or unsupervised manner.

In the unsupervised manner, the value is calculated by:

```
log2(var(GS_\-i)/var(GS))
```

where GS is the gene set score, and the GS_\-i is a recalculate of gene set score without i’th feature. var() is the variance.

In the supervised manner, the value is calculated as the F-ratio over a class vector:
\[ \log_2(\text{F(GS}_i)/\text{F(GS)}) \]

Where \( F() \) is the calculation of F-ratio. The unsupervised GIS is encouraged since it works better for most of the cases in practice.

**Value**

An object of class `data.frame` contains three columns. The first column is the feature name, the second columns is the gene influential score. The third columns indicates from where the feature/gene is selected.

**Author(s)**

Chen Meng

**References**

TBA

**See Also**

see `annotate.gs`

**Examples**

```r
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
msa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
    proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
allgs <- colnames(NCI60_4array_supdata[[1]])

# unsupervised measurement
GIS(msa, allgs[1], topN = 5)

# supervised measurement
tissueType <- as.factor(sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\."), "[", 1))
GIS(msa, allgs[1], topN = 5, Fvalue = TRUE, ff = tissueType)

# more PCs to calcualte
GIS(msa, allgs[1], nf = 20, topN = 5, Fvalue = TRUE, ff = tissueType)
```

---

### matpower

**compute the power of a matrix**

**Description**

the power of a matrix

**Usage**

`matpower(x, n, nf = min(dim(x)), tol = 1e-07)`
Arguments

- **x**: a numerical matrix object that the power of which should be calculated.
- **n**: The matrix to the power of.
- **nf**: The number of axes kept in the calculation of SVD and reconstruction.
- **tol**: The tolerance of the axis, singular vectors with singular value lower than tol will be ignored in the reconstruction.

Details

The power of a matrix is calculated in two steps: decompostion step: \(x=UDV'\) and the reconstruction step: \(x^n=U^*D^n*V'\). In the reconstruction, the singular vectors with a singular value more than tol are kept.

Value

A matrix \(x^n\)

Note

Called by the wsvd function.

Author(s)

Chen Meng

See Also

See Also `wsvd`

Examples

```r
set.seed(56)
m <- matrix(rnorm(15), 5, 3)
s <- matpower(m, 2)
s <- matpower(m, -2)
```

Description

Three approaches are supplied in this function, consensus PCA (CPCA), generalized CCA (GCCA) and multiple co-inertia analysis (MCIA).

Usage

```r
mbpca(x, ncomp, method, k = "all", center = TRUE, scale = FALSE, option = "uniform", maxiter = 1000, moa = TRUE, verbose = TRUE, svd.solver = c("svd", "fast.svd", "propack"))
```
Arguments

`x` A list of matrix or data.frame, where rows are variables and columns are samples. The columns among the matrices need to be match but the variables do not need to be.

`ncomp` An integer; the number of components to calculate. To calculate more components requires longer computational time.

`method` A character string could be one of c("globalScore", "blockScore", "blockLoading"). The "globalScore" approach equals consensus PCA; The "blockScore" approach equals generalized canonical correlation analysis (GCCA); The "blockLoading" approach equals multiple co-inertia analysis (MCIA);

`k` The absolute number (if k $\geq 1$) or the proportion (if 0<k<1) of non-zero coefficients for the variable loading vectors. It could be a single value or a vector has the same length as x so the sparsity of individual matrix could be different.

`center` Logical; if the variables should be centered

`scale` Logical; if the variables should be scaled

`option` A charater string could be one of c("lambda1", "inertia", "uniform") to indicate how the different matrices should be normalized. If "lambda1", the matrix is divided by its the first singular value, if "inertia", the matrix is divided by its total inertia (sum of square), if "uniform", none of them would be done.

`maxiter` Integer; Maximum number of iterations in the algorithm

`moa` Logical; whether the output should be converted to an object of class `moa-class`

`verbose` Logical; whether the process (# of PC) should be printed

`svd.solver` A character string could be one of c("svd", "fast.svd", "propack"). The default "fast.svd" has a good compromise between the robustness and speed. "propack" is the fastest but may failed to converge in practice.

Details
details need to update

Value

An object of class `moa-class` (if moa=TRUE) or an list object contains the following elements:

- `tb` - the block scores
- `pb` - the block loadings
- `t` - the global scores
- `w` - the weights of block scores to construct the global scor

Note

no note now

Author(s)

Chen Meng

References

reference need to be updated
See Also

see moa for non-iterative algorithms for multi-block PCA.

Examples

data("NCI60_4arrays")
tumorType <- sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\\."), "+", 1)
colcode <- as.factor(tumorType)
levels(colcode) <- c("red", "green", "blue", "cyan", "orange", "gray25", "brown", "gray75", "pink")
colcode <- as.character(colcode)

moa <- mbpca(NCI60_4arrays, ncomp = 10, k = "all", method = "globalScore", option = "lambda1",
center=TRUE, scale=FALSE)
plot(moa, value="eig", type=2)
r <- bootMbpca(moa, mc.cores = 1, B=6, replace = FALSE, resample = "sample")
moas <- mbpca(NCI60_4arrays, ncomp = 3, k = 0.1, method = "globalScore", option = "lambda1",
center=TRUE, scale=FALSE)

scr <- moaScore(moa)
scrs <- moaScore(moas)
diag(cor(scr[, 1:3], scrs))

layout(matrix(1:2, 1, 2))
plot(scr[, 1:2], col=colcode, pch=20)
legend("topright", legend = unique(tumorType), col=unique(colcode), pch=20)
plot(scr[, 2:3], col=colcode, pch=20)
gap <- moGap(moas, K.max = 12, cluster = "hcl")
gap$nClust

hcl <- hclust(dist(scrs))
cls <- cutree(hcl, k=4)
clsColor <- as.factor(cls)
levels(clsColor) <- c("red", "blue", "orange", "pink")
clsColor <- as.character(clsColor)

heatmap(t(scrs[hcl$order, ]), ColSideColors = colcode[hcl$order], Rowv = NA, Colv=NA)
heatmap(t(scrs[hcl$order, ]), ColSideColors = clsColor[hcl$order], Rowv = NA, Colv=NA)
genes <- moaCoef(moas)
genes$nonZeroCoef$agilent.V1.neg
Description

mgsa class here.

Objects from the Class

Objects can be created by calls of the form `new("mgsa", ...)`.

Slots

- call: call
- moa: Object of class moa
- sup: Object of class moa.sup

Methods

`signature(x = "mgsa", y = "mgsa")` To combine two objects of class "mgsa"

This function could only be used to combine two "mgsa" objects, using "Reduce" function to combine more.

Author(s)

Chen Meng

See Also

moa and moa.sup

Examples

```r
showClass("mgsa")
# library(mgsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
# split gene set annotation into two sets.
sup1 <- lapply(NCI60_4array_supdata, function(x) x[, 1:10])
sup2 <- lapply(NCI60_4array_supdata, function(x) x[, -(1:10)])
# project two sets of annotation
mgsa1 <- mogsa(x = NCI60_4arrays, sup=sup1, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
mgsa2 <- mogsa(x = NCI60_4arrays, sup=sup2, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
# combine two independent mgsa sets
mgsa_comb <- combine(mgsa1, mgsa2)
dim(getmgsa(mgsa1, "fac.scr"))
dim(getmgsa(mgsa2, "fac.scr"))
dim(getmgsa(mgsa_comb, "fac.scr"))
```
Description

Analysis multiple omics data using MFA or STATIS. The input multiple tables are in a form that columns are samples and rows are variables/features.

Usage

\texttt{moa(data, proc.row=\texttt{"center\_ssq1"}, w.data=\texttt{"inertia"}, w.row=NULL, statis=FALSE)}

Arguments

data A list of \texttt{data.frame} or \texttt{matrix} that contains the input data, the columns in all datasets should be samples/observations (which need to be matched) and rows should be variables.

proc.row Preprocessing of rows of datasets, should be one of \texttt{none} - no preprocessing, \texttt{center} - center only, \texttt{center\_ssq1} - center and scale (sum of squared values equals 1), \texttt{center\_ssqN} - center and scale (sum of squared values equals the number of columns), \texttt{center\_ssqNm1} - center and scale (sum of squared values equals the number of columns - 1) MFA corresponds to \texttt{"proc.row=center\_ssq1" and } \texttt{w.data="lambda1"}

w.data The weights of each separate dataset, should be one of \texttt{uniform} - no weighting, \texttt{lambda1} - weighted by the reverse of the first eigenvalue of each individual dataset or \texttt{inertia} - weighted by the reverse of the total inertia. See detail.

w.row If it is not null, it should be a list of positive numerical vectors, the length of which should be the same with the number of rows of each dataset to indicated the weight of rows of datasets.

statis A logical indicates whether STATIS method should be used. See details.

Details

Different methods employs different precessing of row and datasets. For multiple factorial analysis (MFA), the rows of each dataset are first centered and scaled, then each dataset is weighted by the reverse of its first eigenvalue (\texttt{proc.row=center\_ssq1, w.data="lambda1"}). This algorithm does not have a well defined criterion to be optimized (see reference).

If \texttt{statis=TRUE}, the statis algorithm will be used, that is, each dataset will be further weighted so that datasets closer to the overall structure will receive a higher weight.

Value

An object of class \texttt{moa-class}.

Author(s)

Chen Meng
References


See Also

sup.moa, mogsa. More about plot see moa-class.

Examples

```r
# library(mogsa)
# loading data
data(NCI60_4arrays)
# run analysis
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
# plot
# plot eigen value
plot(ana, value = "eig", type = 2)
# plot the normalized (percentage) eigen value
plot(ana, value = "tau", type = 2)
# plotting the observations
colcode <- as.factor(sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\."), "[", 1))
plot(ana, type = 1, value = "obs", col=colcode)
plot(ana, type = 2, value = "obs", col=colcode, data.pch=1:4)
# plot variables/features in each data sets
plot(ana, value = "var", layout=matrix(1:4, 2, 2))
# plot the RV coefficients for the data sets
plot(ana, value = "RV")
```

moa-class

Class "moa"

Description

moa class object

Objects from the Class

Objects can be created by calls of the form new("moa", ...).

Slots

eig: eigen values
tau: The percentage of explained variance by each datasets sparately.
partial.eig: matrix, rows indicate the partial eigenvalues from each data.
eig.vec: a matrix, eigenvectors.
loading: the coordinate of variables/features.
**moa-class**

- **fac.scr**: factor score of observations.
- **partial.fs**: partial factor score.
- **ctr.obs**: contribution of each observation to the total factor score.
- **ctr.var**: contribution of each variable to the total variance.
- **ctr.tab**: contribution of each data to the total variance.
- **RV**: pairwise RV coefficients
- **w.row**: weight of rows
- **w.data**: weight of datasets
- **data**: the original input data
- **tab.dim**: the dimension of each input data
- **call**: call

### Methods

**plot** signature(x = "moa", y = "missing"): Argument "value" should be one of "eig", "tau", "obs", "var" and "RV"

- If value = "eig", the eigenvalue would be plotted as a scree plot. The following arguments could be set:
  - **type=1**: The type of plot to show eigenvalues. (type=1: the eigenvalue are plotted; type=2: partial eigenvalue shown as concatenated bars; type=3: partial eigenvalue shown as bars side by side; type=4: matplot view of eigenvalues, lty need to be set; type=5: the two dimensional plot of partial eigenvalues, axes and pch need to be set in this case.)
  - **axes=NULL**: The axes selected to plot
  - **n=NULL**: Top n eigenvalues to be drawn
  - **tol=1e-5**: The tolerance of eigenvalue, eigenvalues lower than this value will not be shown.
  - **legend=NULL**: A character string as calling legend function
  - **col=NULL**: The color of partial eigenvalues from each data set
  - **lty=1**: The line type used in the matplot, used when type = 4
  - **pch=NULL**: the pch to draw 2D partial eigen plot, when type = 5 used
  - **lg.x="topright"**: The position of legend
  - **lg.y=NULL**: Position argument passed to function "legend"
  - **...**: other arguments passed to functions

- If value = "tau", the same with eig, but in the eigenvalues are scaled to 1

- If value = "obs", the observation space will be shown, the following argument could be set:
  - **type=1**: Which type, see below (for type=1: the center points draw; type=2: the separate factor scores linked by lines; ... will be passed to function "points")
  - **data.pch=20**: the pch of dataset, if type=1, the first one is used
  - **col=1**: the color of observations, recycled used by data.frame
  - **label=FALSE**: A logical indicates if labels should be shown
  - **lg.x="topright"**: Position of legend
  - **lg.y=NULL**: Position of legend
  - **xlim=NULL**: The x limit
  - **ylim=NULL**: The y limit
  - **label.cex=1**: the cex of text

- **var**: the separate gene view, layout can be specified
- **RV**: the heatmap of RV coefficients
Author(s)
Chen Meng

References
Herve Abdi, Lynne J. Williams, Domininique Valentin. Multiple factor analysis: principal component analysis for multitable and multiblock data sets. WIREs Comput Stat 2013

Examples

```r
tshowClass("moa")
# load("R/mogsa/data/NCI60_4arrays.rda")
data(NCI60_4arrays)
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)

plot(ana, value="eig")
plot(ana, value="tau", type=2)
```

Description
moa.sup class desc.

Objects from the Class
Objects can be created by calls of the form `new("moa.sup", ...)`.

Slots
- `sup`: Object of class "list", the matrix of supplementary data.
- `coord.sep`: The projection of geneset information on each separate data.
- `coord.comb`: The projection of geneset information on total dataset.
- `score`: the gene set-sample pathway score
- `score.data`: the gene set-sample pathway score, data separate
- `score.pc`: the gene set-sample pathway score, PC separate
- `score.sep`: the gene set-sample pathway score, separate.
- `p.val`: the p value matrix have the same dimension with score matrix.

Methods
There is no generic function for objects of "moa.sup", but have specific function, including: - decompose.gs.ind - box.gs.feature - plotGS - decompose.gs.group
moaCoef

Author(s)
Chen Meng

See Also
objects to See Also as decompose.gs.ind, box.gs.feature, plotGS, decompose.gs.group.

Examples
showClass("moa.sup")
data(NCI60_4array_supdata)
data(NCI60_4arrays)
sapply(NCI60_4array_supdata, dim)
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
plot(ana, value="eig")
smoa <- sup.moa(ana, sup=NCI60_4array_supdata, nf=5)

moaCoef(moa)

Arguments
moa An object of class moa-class.

Value
It returns a list consist of two components:
coefMat - the loading matrix
nonZeroCoef - it is a list of data.frame to list the non-zero coefficient variable in each of loading vectors and data sets. The element names are in a format as "xxxx.yy.zzz"
xxxx - are the data names, tells the data set where a varirable is from
yy - the number of Axes, for example, "V1" indicate the variable has a non-zero coefficient in the first loading vector.
zzz - could be either "pos" (coefficient >0) or "neg" (coefficient < 0)
The data.frame has two columns, the first column is the ID of a variable the second column is the coefficient/loading.

Author(s)
Chen Meng
moaScore

See Also

moaScore

Examples

# see examples in \code{\link{mbpca}}

data("NCI60_4arrays")
moa <- mbpca(NCI60_4arrays, ncomp = 10, k = "all", method = "globalScore", option = "lambda1",
            center=TRUE, scale=FALSE)

genes <- moaCoef(moa)
scr <- moaScore(moa)

moaScore Extract global scores from an object of class moa-class.

Description

Extract global scores from an object of class moa-class.

Usage

moaScore(moa)

Arguments

moa An object of class moa-class

Value

A matrix of global score

Author(s)

Chen Meng

See Also

moaCoef

Examples

# see examples in \code{\link{mbpca}}

data("NCI60_4arrays")
moa <- mbpca(NCI60_4arrays, ncomp = 10, k = "all", method = "globalScore", option = "lambda1",
            center=TRUE, scale=FALSE)

genes <- moaCoef(moa)
scr <- moaScore(moa)
moGap

Gap statistic for clustering latent variables in moa-class.

Description
Gap statistic is a measurement of goodness of clustering result. This is a convenient function to calculate the gap statistic of clustering "moa".

Usage
moGap(x, K.max, B = 100, cluster = c("kmeans", "hclust"), plot = TRUE, dist.method = "euclidean", dist.diag = FALSE, dist.upper = FALSE, dist.p = 2, hcl.method = "complete", hcl.members = NULL, km.iter.max = 10, km.nstart = 10, km.algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), km.trace = FALSE)

Arguments
- x: An object of class moa-class returned by mbpca.
- K.max: The maximum number of clusters to consider, passed to clusGap
- B: The number of bootstrap, passed to clusGap
- cluster: A character string could be either "kmeans" or "hclust" to specify the clustering algorithm.
- plot: Logical; whether return the gap statistic plot.
- dist.method: Distance measurement, passed to function "dist".
- dist.diag: Passed to function "dist".
- dist.upper: Passed to function "dist".
- dist.p: Passed to function "dist".
- hcl.method: Hierarchical clustering method, passed to "hclust"
- hcl.members: Passed to "hclust"
- km.iter.max: Maximum number of iteration in kmeans, passed to "kmeans".
- km.nstart: An integer to specify how many random sets should be chosen. passed to "kmeans".
- km.algorithm: Kmeans algorithm, passed to "kmeans".
- km.trace: See function "kmeans".

Value
It returns a list consists of five components:
"Tab", "n", "B", "FUNcluster" - see clusGap
"nClust" - the estimated number of clusters using different method, see maxSE

Author(s)
Chen Meng
References


See Also

Function "clusGap" in "cluster" package Function "dist", "hclust", "kmeans"

Examples

# see examples in \code{\link{mbpca}}

data("NCI60_4arrays")
moa <- mbpca(NCI60_4arrays, ncomp = 10, k = "all", method = "globalScore", option = "lambda1",
center=TRUE, scale=FALSE)
gap <- moGap(moa, K.max = 12, cluster = "hcl")
gen <- moaCoef(moa)
scr <- moaScore(moa)

mogsa

multiple omics data integration and gene set analysis

Description

The main function called by users, omics data analysis and gene set annotation. A wrapper function of moa and sup.moa.

Usage

mogsa(x, sup, nf=NULL, proc.row=NULL, w.data=NULL, w.row=NULL, statis=FALSE, ks.stat=FALSE, ks.B = 1000, ks.cores = NULL)

Arguments

x An object of class list or moa-class. A list would be a list of data frame.
sup An object of class list or moa.sup-class. A list would be a list of supplementary data.
nf The number of principal components used to reconstruct, only used when x is a an object of list.
proc.row Preprocessing of rows. If x is a object of list, it is passed moa
w.data Weights of datasets. If x is a object of list, it is passed moa
w.row Weight of row. If x is a object of list, it is passed moa
statis A logical indicates if statis algorithm should be used. If x is a object of list, it is passed moa
ks.stat  The logical indicates if the p-value should be calculated using K-S statistic (the method used in "ssgsea" in GSVA package). Default is FALSE, which means using the z-score method. See sup.moa.

ks.B  An integer to indicate the number of bootstrapping samples to calculated the p-value of KS statistic.

ks.cores  An integer indicate the number of cores to be used in bootstrapping. It is passed to function mclapply in the parallel package.

Details

A wrapper function of moa and sup.moa.

Value

An object of class mgsa-class.

Note

This function will be changed to a generic function for "S4-style" programming.

Author(s)

Chen Meng

References


See Also

moa and sup.moa

Examples

```r
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)

# using a list of data.frame as input
mgsa1 <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
# using moa as input
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
smoa <- sup.moa(ana, sup=NCI60_4array_supdata, nf=3)
mgsa2 <- mogsa(x = ana, sup=NCI60_4array_supdata, nf=9)
mgsa3 <- mogsa(x = ana, sup=smoa)
```
## msvd

**SVD based algorithm to calculate block Score and global scores for mbpca.**

### Description

An internal function called by `mbpca`. It returns the result comparable with `nipalsSoftK`, but way faster since it uses the SVD algorithm. No sparse operators in this function.

### Usage

```r
msvd(x, svd.sol = svd)
```

### Arguments

- **x**: The input matrix, rows are observations, columns are variables
- **svd.sol**: A function object to specify the preferred SVD solver, default is `svd`.

### Value

A list object contains the following elements:

- `tb`: the block scores
- `pb`: the block loadings
- `t`: the global scores
- `w`: the weights of block scores to construct the global scores

### Author(s)

Chen Meng

### See Also

- `nipalsSoftK`

---

## NCI60_4arrays

### Microarray gene expression profiles of the NCI 60 cell lines from 4 different platforms

### Description

The 60 human tumour cell lines are derived from patients with leukaemia, melanoma, lung, colon, central nervous system, ovarian, renal, breast and prostate cancers. The cell line panel is widely used in anti-cancer drug screen. In this dataset, a subset of microarray gene expression of the NCI 60 cell lines from four different platforms are combined in a list, which could be used as input to `mcia` directly.

### Usage

```r
data(NCI60_4arrays)
```
Format

The format is: List of 4 data.frames

- \$agilent: data.frame containing 300 rows and 60 columns. 300 gene expression log ratio measurements of the NCI60 cell lines, by Agilent platform.
- \$hgu133: data.frame containing 298 rows and 60 columns. 298 gene expression log ratio measurements of the NCI60 cell lines, by H-GU133 platform.
- \$hgu133p2: data.frame containing 268 rows and 60 columns. 268 gene expression log ratio measurements of the NCI60 cell lines, by H-GU133 plus 2.0 platform.
- \$hgu95: data.frame containing 288 rows and 60 columns. 288 gene expression log ratio measurements of the NCI60 cell lines, by H-GU95 platform.

Value

NCI60_4arrays will be loaded in your working space.

Source

Cell Miner http://discover.nci.nih.gov/cellminer/

References


NCI60_4array_supdata supp data for Microarray gene expression profiles of the NCI 60 cell lines from 4 different platforms

Description

Supplementary to NCI60_4arrays.

Usage

data(NCI60_4arrays)

Format

The format is: List of 4 matrix

- \$agilent: matrix containing 300 rows and 60 columns. 300 gene expression log ratio measurements of the NCI60 cell lines, by Agilent platform.
- \$hgu133: matrix containing 298 rows and 60 columns. 298 gene expression log ratio measurements of the NCI60 cell lines, by H-GU133 platform.
- \$hgu133p2: matrix containing 268 rows and 60 columns. 268 gene expression log ratio measurements of the NCI60 cell lines, by H-GU133 plus 2.0 platform.
- \$hgu95: matrix containing 288 rows and 60 columns. 288 gene expression log ratio measurements of the NCI60 cell lines, by H-GU95 platform.
 nipalsSoftK

**Value**

NCI60_array_supdata will be loaded in your working space.

---

| nipalsSoftK | NIPALS algorithm with soft thresholding operator |

**Description**

An internal function called by mbpca.

**Usage**

`nipalsSoftK(x, maxiter, k)`

**Arguments**

- `x` The input matrix, rows are observations, columns are variables
- `maxiter` # of maximum interation the algorithm can run
- `k` The number (>=1) or proportion (<1) of variables want to keep. It could be a single value or a vector has the same length as x so the sparsity of individual matrix could be different.

**Value**

an list object contains the following elements:

- `tb` - the block scores
- `pb` - the block loadings
- `t` - the global scores
- `w` - the weights of block scores to construct the global score.

**Author(s)**

Chen Meng

**See Also**

msvd
pairwise.rv

pairwise.rv  pairwise RV coefficients.

Description
Calculating pairwise RV coefficients for a list of matrices or data.frame.

Usage
pairwise.rv(data.list, match="col")

Arguments
- data.list: A list of data.frame or matrix, either rows or columns in each data set should be matched.
- match: Whether columns or rows of data.frame/matrix should be matched.

Details
The RV coefficient for each pair of matrices is calculated as $R_v = \text{trace}(XX'YY')/\sqrt{\text{trace}(XX'XX')\text{trace}(YY'YY')}$

Value
The function will return a matrix containing the pairwise RV coefficients.

Note
The variable in matrices are not automatically centered or scaled in this function. So these step may need to be performed before calling this function.

Author(s)
Chen Meng

References

Examples
data(NCI60_4arrays)
pairwise.rv(NCI60_4arrays)
plot-methods

Description

Methods

signature(x = "moa", y = "missing") plot moa object

Argument "value" could be one of "eig", "tau", "obs", "var" and "RV"

if value = "eig", the eigenvalue would be plotted as scree plot. The following arguments could be set:

- type=1 - The type of plot to show eigenvalues. (type=1: the eigenvalue are plotted; type=2: partial eigenvalue shown as concatenated bars; type=3: partial eigenvalue shown as bars side by side; type=4: matplot view of eigenvalues, lty need to be set; type=5: the two dimensional plot of partial eigenvalues, axes and pch need to be set in this case.)
- axes=NULL - The axes selected to plot
- n=NULL - Top n eigenvalues to be drawn
- tol=1e-5 - The tolerance of eigenvalue, eigenvalues lower than this value will not be shown.
- legend=NULL - legend to put, a character string as calling legend function
- col=NULL - The color of partial eigenvalues from each data set
- lty=1 - The line type used in the matplot, used when type =4 \ pch= NULL - the pch to draw 2D partial eigen plot, when type = 5 used \ lg.x="topright" - The position of legend \ lg.y= NULL - Poistion argument passed to function "legend" \ ... - other arguemnts passed to functions

if value = "tau", the same with eig, but in the eigenvalues are scaled to 1

if value = "obs", the observation space will be shown, the following argument could be set:

- axes=1:2 - Which axes should be draw\type=1 - Which type, see below (for type=1: the center points draw; type=2: the separate factor scores linked by lines; ... will be passed to function "points")
- data.pch=20 - the pch of dataset, if type=1, the first one is used\data.col=1 - the color of observations, recycled used by data.frame\label=FALSE - A logical indicates if labels should be shown\lg.x="topright" - Position of legend \lg.y= NULL - Position of legend \xlim= NULL - The x limit \ylim= NULL - The y limit \label.cex=1 - the cex of text \...

var - the separate gene view, layout can be specified

RV - the heatmap of RV coefficients

plotGS

Plot the gene set space

Description

Plot the gene set space of objects of "moa" and "mgsa"

Usage

plotGS(x, axes=1:2, center.only=FALSE, topN=1, data.pch=20, data.col=1, highlight.col = 2, label=NULL, label.cex=1, layout=NULL, ...)
Arguments

- **x**: An object of class `mgsa-class` or `moa.sup-class`.
- **axes**: An integer vector in the length 2 to indicate the axes to be drawn.
- **center.only**: A logical to indicate whether the separate gene set spaces from each of the data set should be plotted. Default is FALSE.
- **topN**: An integer specify N gene set from the most positive and negative end of axes to be labeled.
- **data.pch**: The shape for plotting each data set. This argument is passed to `points` function, so only used when separate gene set spaces are plotted (i.e. `center.only = FALSE`).
- **data.col**: The col for plotting each data set. This argument is passed to `points` function, so only used when separate gene set spaces are plotted (i.e. `center.only = FALSE`).
- **highlight.col**: The color used to highlight the selected gene sets.
- **label**: Either a character vector or NULL (default). The character vector should be the name of some gene sets want ot be labeled.
- **label.cex**: Passed to `text` function to adjust the the labels.
- **layout**: A matrix passed to the `layout` function.
- **...**: Other arguments passed to `points`.

Details

This is a convenience function to explore the gene set space so not very flexible. For customized plot, please use the object of `data@coord.comb` and `data@coord.sep`.

Value

If assign to variable, A list of selected/highlighted gene set at the (positive and negative) end of each axis will be returned.

Author(s)

Chen Meng

Examples

```r
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
mgsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
              proc.row = "center_ssq!", w.data = "inertia", statis = TRUE)

plotGS(mgsa, center.only = TRUE, topN=5)
res <- plotGS(mgsa, center.only = FALSE, data.pch=1:4, data.col=1:4)
res
```
prepGraphite

Description

Prepare pathway gene sets from "graphite" package, which could be passed to "prepSupMoa" function.

Usage

prepGraphite(db, id = c("entrez", "symbol"))

Arguments

db The database to be used, an object of class either 'PathwayList' create by "pathways" function.

id Which identifier for output, either "entrez" or "symbol".

Details

Only support "entrez" or "symbol" output currently.

Value

This function returns an object of list containing gene set information, which could be further processed by function "prepSupMoa" to convert to the object that can be used as input of "sup.moa" or "mogsa".

Author(s)

Chen Meng

References


See Also

See Also as prepMsigDB and prepSupMoa.

Examples

library(graphite)
keggdb <- prepGraphite(db = pathways("hsapiens", "kegg")[,3], id = "entrez")


**Description**

Convert a gmt file (Could be downloaded from MSigDB) to a list of gene sets information.

**Usage**

```r
prepMsigDB(file)
```

**Arguments**

- `file` The directory and file name of the gmt file.

**Value**

This function returns an object of list containing gene set information, which could be further processed by function `prepSupMoa` to convert to the object that can be used as input of "sup.moa" or "mogsa".

**Author(s)**

Chen Meng

**See Also**

See Also as `prepGraphite` and `prepSupMoa`.

**Examples**

```r
# not run
dir <- system.file(package = "mogsa")
preGS <- prepMsigDB(file=paste(dir,
"/extdata/example_msigdb_data.gmt.gz", sep = ""))
```

---

**Description**

Prepare supplementary tables for projection by sup.moa or mogsa.

**Usage**

```r
prepSupMoa(X, geneSets, minMatch = 10, maxMatch = 500)
```
processOpt

Arguments

\(X\)  
A matrix/data.frame or a list of matrix/data.frame or a list of character vector. If it is a list of matrix/data.frame, row names of matrix/data.frame will be used to create the projection matrix. Otherwise the character vectors will used to create the supplementary matrix.

geneSets  
Gene sets list or an object of class "GeneSet" or "GeneSetCollection". A gene set list could be returned by prepGraphite or prepMolsigDB.

minMatch  
The minimum match of geneset.

maxMatch  
The maximum match genesets.

Details

Details here

Value

A list of matrix could used as supplementary tables by "sup.moa" or "mogsa".

Author(s)

Chen Meng

See Also

See Also as prepGraphite and prepMsigDB.

Examples

```r
library(graphite)
data(NCI60_4arrays)
gss <- prepGraphite(db = kegg[6:10], id="symbol")
sup_data1 <- prepSupMoa(NCI60_4arrays, geneSets=gss)
gene_list <- lapply(NCI60_4arrays, rownames)
sup_data2 <- prepSupMoa(gene_list, geneSets=gss)
```

Description

An internal function called by mbpca.

Usage

```
processOpt(x, center = TRUE, scale = FALSE, option = c("lambda1", "inertia", "uniform"))
```
**Arguments**

- **x**: A list of matrices, rows are observations and columns are variables
- **center**: A logical variable indicates whether columns should be centered
- **scale**: A logical variable indicates whether columns should be scaled
- **option**: A character string could be one of c("lambda1", "inertia", "uniform") to indicate how the different matrices should be normalized. If "lambda1", the matrix is divided by its the first singular value, if "inertia", the matrix is divided by its total inertia (sum of square), if "uniform", none of them would be done.

**Value**

A list of normalized matrix.

**Author(s)**

Chen Meng

---

**softK**

*Soft-thresholding operator*

**Description**

Soft-thresholding operator, which is called by `mbpca`.

**Usage**

`softK(x, k)`

**Arguments**

- **x**: A numerical vector
- **k**: Number of non-zero elements want to keep

**Value**

A numerical vector

**Author(s)**

Chen Meng

**Examples**

```r
v <- rnorm(10)
softK(v, k = 2)
```
sup.moa

Projecting supplementary tables on object of class moa-class.

Description

Projecting supplementary tables on moa-class

Usage

sup.moa(X, sup, nf = 2, ks.stat=FALSE, ks.B = 1000, ks.cores = NULL)

Arguments

X
An object of class moa-class

sup
A list of data.frames contains supplementary data.

nf
The number of principal components used in the projection.

ks.stat
The logical indicates if the p-value should be calculated using K-S statistic (the
method used in "ssgsea" in GSVA package). Default is FALSE, which means
using the z-score method.

ks.B
An integer to indicate the number of bootstrapping samples to calculated the
p-value of KS statistic.

ks.cores
An integer indicate the number of cores to be used in bootstrapping. It is passed
to function mclapply in the parallel package.

Details

Projecting supplementary tables on moa-class, for details see reference.

Value

An object of class moa.sup-class.

Author(s)

Chen Meng

References

Herve Abdi, Lynne J. Williams, Domininique Valentin and Mohammed Bennani-Dosse. STATIS
and DISTATIS: optimum multitable principal component analysis and three way metric multidimen-
Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data.
Examples

```
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
# check the dimension of each supplementary data to see how many gene set annotated the data
sapply(NCI60_4array_supdata, dim)
# run analysis
ana <- moa(NCI60_4arrays, proc.row = "center_ssq!", w.data = "inertia", statis = TRUE)
plot(ana, value="eig")
# projectin supplementary data
smoa <- sup.moa(ana, sup=NCI60_4array_supdata, nf=3)
# heatmap visualize the gene set scores
heatmap(slot(smoa, "score"))
```

toMoa

`convert mbpca result to moa-class`

Description

An internal function called by `mbpca`.

Usage

toMoa(data, x, call)

Arguments

data
  The preprocessed data in `mbpca`

x
  The object calculated in `mbpca`

call
  The call of `mbpca`

Value

An object of `moa-class`.

Author(s)

Chen Meng
**Description**

The weighted version of singular value decomposition.

**Usage**

```r
wsvd(X, D1 = diag(1, nrow(X)), D2 = diag(1, ncol(X)))
```

**Arguments**

- **X**: A numeric matrix whose wSVD decomposition is to be computed.
- **D1**: A square matrix or vector. The left constraint/weight matrix (symmetric and positive in diagonal). The dimension of D1 should be the same with the number of rows in X. A vector input will be converted to a diagonal matrix.
- **D2**: A square matrix or vector. The right constraint/weight matrix (symmetric, positive in diagonal). The dimension of D2 should be the same with the number of columns in X. A vector input will be converted to a diagonal matrix.

**Details**

The weighted version of generalized singular value decomposition (SVD) of matrix \( A = UDV' \) with the constraints \( U'D1U = I \) and \( V'D2V = I \). D1 and D2 are two matrices express constraints imposed on the rows and the columns of matrix A.

**Value**

- `d` - singular values
- `u` - left singular vectors
- `v` - right singular vectors
- `D1` - the left weight matrix (directly from input)
- `D2` - the right weight matrix (directly from input)

**Author(s)**

Chen Meng

**References**


**See Also**

`svd`
Examples

```r
set.seed(56)
m <- matrix(rnorm(15), 5, 3)
w1 <- rnorm(5)
wr <- runif(3)
s <- wsvd(X=m, D1=w1, D2=wr)
# t(s$u) %*% diag(w1) %*% s$u
# t(s$v) %*% diag(wr) %*% s$v
# all.equal(m, as.matrix(s$u) %*% diag(s$d) %*% t(s$v))
```
Index

*Topic CPCA
  mbpca, 15
*Topic GCCA
  mbpca, 15
*Topic MCIA
  mbpca, 15
*Topic MFA
  moa, 19
*Topic MVA
  moa, 19
  mogsa, 26
*Topic Microarray
  NCI60_4array_supdata, 29
  NCI60_4arrays, 28
*Topic NCI-60
  NCI60_4array_supdata, 29
  NCI60_4arrays, 28
*Topic PCA
  moa, 19
*Topic RV coefficient
  pairwise.rv, 31
*Topic STATIS
  moa, 19
*Topic SVD
  wsvd, 40
*Topic classes
  mgsa-class, 17
  moa-class, 20
  moa.sup-class, 22
*Topic combine
  combine-methods, 7
*Topic data projection
  sup.moa, 38
*Topic datasets
  NCI60_4array_supdata, 29
  NCI60_4arrays, 28
*Topic gap statistic
  moGap, 25
*Topic generalized SVD
  wsvd, 40
*Topic graphite
  prepGraphite, 34
*Topic methods
  plot-methods, 32
*Topic mgsa-class
  combine-methods, 7
*Topic moa-class
  plot-methods, 32
*Topic moa
  moGap, 25
*Topic mogsa
  combine-methods, 7
*Topic multi-block PCA
  mbpca, 15
*Topic pahtways
  prepGraphite, 34
*Topic soft threshold
  softK, 37
*Topic supplementary data projection
  mogsa, 26
*Topic supplementary data
  sup.moa, 38
*Topic weighted SVD
  wsvd, 40
annotate.gs, 3, 14
bootMbpca, 4, 5, 6
bootMbpcaK, 5
box.gs.feature, 6, 23
boxplot, 6
combine (combine-methods), 7
combine, mgsa, mgsa-method (combine-methods), 7
combine-methods, 7
de decompose.gs.group, 8, 10, 23
de decompose.gs.ind, 9, 9, 23
deflat, 10
distMoa, 11
getmgsa, 12
GIS, 4, 13
matpower, 14
mbpca, 4, 5, 10, 15, 25, 28, 30, 36, 37, 39
mgsa-class, 17
INDEX

moa, 4, 6, 17, 18, 19, 26, 27
moa-class, 11, 20, 23–25, 39
moa.sup, 18
moa.sup-class, 22
moaCoef, 23, 24
moaScore, 24, 24
moGap, 25
mogsa, 20, 26
mogsa-package, 2
msvd, 11, 28, 30

NCI60_4array_supdata, 29
NCI60_4arrays, 28
nipalsSoftK, 11, 28, 30

pairwise.rv, 31
plot, moa, missing-method (moa-class), 20
plot-methods, 32
plotGS, 23, 32
points, 33
prepGraphite, 34, 35, 36
prepMsigDB, 34, 35, 36
prepSupMoa, 34, 35, 35
processOpt, 36

slot, 12
softK, 37
sup.moa, 20, 26, 27, 38

text, 33
toMoa, 39

wsvd, 15, 40