

# Package ‘mogsa’

October 9, 2015

**Type** Package

**Title** Multiple omics data integration and gene set analysis

**Version** 1.0.3

**Date** 2015-07-23

**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

**VignetteBuilder** knitr

**Suggests** BiocStyle, knitr

**biocViews** GeneExpression, PrincipalComponent, StatisticalMethod, Software

**NeedsCompilation** no

## R topics documented:

mogsa-package	2
annotate.gs	3
box.gs.feature	4
combine-methods	5
decompose.gs.group	6
decompose.gs.ind	7
getmgsa	9
GIS	10
matpower	12
mgsa-class	13
moa	14

moa-class . . . . .	15
moa.sup-class . . . . .	17
mogsa . . . . .	18
NCI60_4arrays . . . . .	20
NCI60_4array_supdata . . . . .	21
pairwise.rv . . . . .	21
plot-methods . . . . .	22
plotGS . . . . .	23
prepGraphite . . . . .	24
prepMsigDB . . . . .	25
prepSupMoa . . . . .	26
sup.moa . . . . .	27
wsvd . . . . .	28

<b>Index</b>	<b>30</b>
--------------	-----------

---

mogsa-package	<i>Multiple omics gene set analysis</i>
---------------	---

---

## Description

Modern "omics" technologies enable quantitative monitoring of the abundance of various biological molecules in a high-throughput manner, accumulating an unprecedented amount of quantitative information on a genomic scale. Gene set analysis is a particularly useful method in high throughput data analysis since it can summarize single gene level information into the biological informative gene set levels. This package provide a method do the gene set analysis based on multiple omics data that describing the same set of observations/samples.

## Details

Package:	mogsa
Type:	Package
Version:	0.1
Date:	2014-06-07
License:	GPL-2
Depends:	methods

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

TBA

## 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", stasis = TRUE)
# using moa as input
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", stasis = 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)
```

---

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 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 <code>mgsa-class</code> or <code>moa.sup-class</code> .
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**

```
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
mogsa <- 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]])
annotate.gs(mogsa, allgs[1])
```

---

box.gs.feature      *boxplot of gene set variables across all samples.*

---

**Description**

boxplot to show the variables (e.g. gene expression) of a gene set 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 <code>mogsa-class</code> or <code>moa.sup-class</code>
gs	Gene set want to be explored
moa	An object of class <code>moa</code> . It is required if x is an object of class <code>moa.sup-class</code>
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 <code>boxplot</code>

**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)
mogsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
              proc.row = "center_ssq1", w.data = "inertia", stasis = TRUE)

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

---

combine-methods

*Combine two objects of class mogsa into one.*

---

## Description

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

## Usage

```
combine(x, y, ...)
```

## Arguments

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

## Value

A combined object of class mogsa will be returned.

## Methods

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

This function could only be used to combine two "mogsa" 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
mgsa1 <- mogsa(x = NCI60_4arrays, sup=sup1, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", stasis = TRUE)
mgsa2 <- mogsa(x = NCI60_4arrays, sup=sup2, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", stasis = 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	<i>Data-wise or PC-wise decomposition of gene set scores for all observations.</i>
--------------------	--

---

**Description**

Data-wise or PC-wise decomposition of gene set scores (GSS) across all observations. The predefined 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, ...)
```

**Arguments**

x	An object of class <a href="#">mgsa-class</a> or <a href="#">moa.sup-class</a>
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 charater 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.
...	Other arguments passed to <code>barplot</code> .

## 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.

## Author(s)

Chen Meng

## References

TBA

## See Also

See Also [decompose.gs.ind](#)

## 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
mogsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
              proc.row = "center_ssq1", w.data = "inertia", stasis = TRUE)

colcode <- as.factor(sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\\.\\.\\."), "[", 1))
decompose.gs.group(x = mogsa, gs = 2, group = colcode, decomp = "data", plot = TRUE)
decompose.gs.group(x = mogsa, gs = 2, group = colcode, decomp = "pc", nf = 3, plot = TRUE)
```

---

<code>decompose.gs.ind</code>	<i>Data-wise or PC-wise decomposition of gene set scores for a single observation.</i>
-------------------------------	--

---

## Description

Barplot of decomposed gene set scores, either with respect to datasets or axes.

## Usage

```
decompose.gs.ind(x, gs, obs, type = 3, nf = 2, plot=TRUE, col.data = NULL,
                col.pc = NULL, legend = TRUE)
```

## Arguments

x	An object of class <a href="#">mgsa-class</a> or <a href="#">moa.sup-class</a>
gs	The gene set want to exam.
obs	The observations want to exam.
type	Which type of plot. type=1 - the data-pc mode; type=2 - the pc-data mode; type=3 - both. See detail.
nf	The number of axes/PCs to be calculated and plotted.
plot	A logical indicates if a plot should be drawn
col.data	The bar color of datasets
col.pc	The bar color of PCs
legend	A logical if legend should be shown

## 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

```
# 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_ssq1", w.data = "inertia", stasis = 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)
```

---

getmgsa	<i>get values in an object of class "mgsa".</i>
---------	---

---

### Description

get values/slot in an object of class "mgsa". The "mgsa" consists of two S4 class objects, [moa-class](#) and [moa.sup-class](#). This function could extract values in these two components directly.

### Usage

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

### Arguments

mgsa	An object of class <a href="#">mgsa-class</a> .
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 [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 extact corresponding value from [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 [moa.sup-class](#).

### Value

The function return the selected value in "mgsa".

### Author(s)

Chen Meng

### References

TBA

## Examples

```
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
mogsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
part.eig <- getmogsa(mogsa, "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

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

## Arguments

x	An object of class <code>mgsa-class</code> .
geneSet	A character string or number to indicate 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 be 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.

## 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:

$$\log_2(\text{var}(\text{GS}_{-i})/\text{var}(\text{GS}))$$

where GS is the gene set score, and the GS<sub>-i</sub> 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:

$$\log_2(F(GS_{-i})/F(GS))$$

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 column is the gene influential score. The third column indicates from where the feature/gene is selected.

### Author(s)

Chen Meng

### References

TBA

### See Also

see [annotate.gs](#)

### 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_ssq1", w.data = "inertia", statis = TRUE)
allgs <- colnames(NCI60_4array_supdata[[1]])

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

# supervised measurement
tissueType <- as.factor(sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\\.\\.\\."), "[", 1))
GIS(mgsa, allgs[1], topN = 5, Fvalue = TRUE, ff = tissueType)
# more PCs to calculate
GIS(mgsa, 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: decomposition 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**

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

---

mgsa-class	Class "mgsa"
------------	--------------

---

### 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

```
showClass("mgsa")
# 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
mgsa1 <- mgsa(x = NCI60_4arrays, sup=sup1, nf=9,
             proc.row = "center_ssq1", w.data = "inertia", stasis = TRUE)
mgsa2 <- mgsa(x = NCI60_4arrays, sup=sup2, nf=9,
             proc.row = "center_ssq1", w.data = "inertia", stasis = 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"))
```

moa

*Multiple omics data analysis using MFA or STATIS***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**

```
moa(data, proc.row="center_ssq1", w.data="inertia", w.row=NULL, stasis=FALSE)
```

**Arguments**

<code>data</code>	A list of data.frame or matrix that contains the input datas, the columns in all datasets should be samples/observations (which need to be matched) and rows should be variables.
<code>proc.row</code>	Preprocessing of rows of datasets, should be one of none - no preprocessing, center - center only, center_ssq1 - center and scale (sum of squared values equals 1), center_ssqN - center and scale (sum of squared values equals the number of columns), center_ssqNm1 - center and scale (sum of squared values equals the number of columns - 1)
<code>w.data</code>	The weights of each separate dataset, should be one of uniform - no weighting, lambda1 - weighted by the reverse of the first eigenvalue of each individual dataset or inertia - weighted by the reverse of the total inertia. See detail.
<code>w.row</code>	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.
<code>stasis</code>	A logical indicates whether STATIS method should be used. See details.

**Details**

Different methods employs different preprocessing 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 (`proc.row=center_ssq1, w.data="lambda1"`). This algorithm does not have a well defined criterion to be optimized (see reference).

If `stasis=TRUE`, the stasis 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 `moa-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 multidimensional scaling. WIREs Comput Stat 2012. Volume 4, Issue 2, pages 124-167

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

**See Also**

[sup.moa](#), [mogsa](#). More about plot see [moa-class](#).

**Examples**

```
# library(mogsa)
# loading data
data(NCI60_4arrays)
# run analysis
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", stasis = 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.  
**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 variables to the total variance.  
**ctr.tab:** contribution of each data to the total variance.  
**RV:** pairwise RV coefficients  
**w.row:** weight of rows  
**data:** the original input data  
**tab.dim:** the dimension of each input data  
**call:** call

**Methods**

**plot** signature(x = "moa", y = "missing"): 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 - 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:  
 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  
 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, Dominique Valentin and Mohammed Bennani-Dosse. STATIS and DISTATIS: optimum multitable principal component analysis and three way metric multidimensional scaling. WIREs Comput Stat 2012. Volume 4, Issue 2, pages 124-167

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

### Examples

```

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

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

```

---

moa.sup-class	Class "moa.sup"
---------------	-----------------

---

### 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`

## 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)
```

---

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)
```

**Arguments**

<code>x</code>	An object of class <code>list</code> or <a href="#">moa-class</a> . A list would be a list of data frame.
<code>sup</code>	An object of class <code>list</code> or <a href="#">moa.sup-class</a> . A list would be a list of supplementary data.
<code>nf</code>	The number of principal components used to reconstruct, only used when <code>x</code> is a an object of <code>list</code> .
<code>proc.row</code>	Preprocessing of rows. If <code>x</code> is a object of <code>list</code> , it is passed <code>moa</code>
<code>w.data</code>	Weights of datasets. If <code>x</code> is a object of <code>list</code> , it is passed <code>moa</code>
<code>w.row</code>	Weight of row. If <code>x</code> is a object of <code>list</code> , it is passed <code>moa</code>
<code>statis</code>	A logical indicates if statis algorithm should be used. If <code>x</code> is a object of <code>list</code> , it is passed <code>moa</code>

**Details**

A wrapper function of [moa](#) and [sup.moa](#).

**Value**

An object of class [mogsa-class](#).

**Note**

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

**Author(s)**

Chen Meng

**References**

Not available right now.

**See Also**

[moa](#) and [sup.moa](#)

**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
mogsa1 <- 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)
```

---

NCI60_4arrays	<i>Microarray gene expression profiles of the NCI 60 cell lines from 4 different platforms</i>
---------------	--

---

## 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

```
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

Reinhold WC, Sunshine M, Liu H, Varma S, Kohn KW, Morris J, Doroshow J, Pommier Y CellMiner: A Web-Based Suite of Genomic and Pharmacologic Tools to Explore Transcript and Drug Patterns in the NCI-60 Cell Line Set. *Cancer Research*. 2012 Jul, 15;72(14):3499-511

---

NCI60\_4array\_supdata *supp data for Microarray gene expression profiles of the NCI 60 cell lines from 4 different platforms*

---

**Description**

Supplimentary 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.

**Value**

NCI60\_4array\_supdata will be loaded in your working space.

---

`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**

<code>data.list</code>	A list of data.frame or matrix, either rows or columns in each data set should be matched.
<code>match</code>	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**

Robert, P.; Escoufier, Y. (1976). A Unifying Tool for Linear Multivariate Statistical Methods: The RV-Coefficient. *Applied Statistics* 25 (3): 257-265.

**Examples**

```
data(NCI60_4arrays)
pairwise.rv(NCI60_4arrays)
```

---

plot-methods

~~ *Methods for Function plot* ~~

---

**Description**

~~ Methods for function plot ~~

**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 - 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:\

axes=1:2 - Which axes should be drawn \ type=1 - Which type, see below (for type=1: the center points drawn; 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

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 <code>mgsa-class</code> or <code>moa.sup-class</code>
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 to be labeled.
label.cex	Passed to <code>text</code> function to adjust the the labels
layout	A matrix passed to the layout function.
...	Other arguments passed to <code>points</code>

**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**

```
# library(mogsa)
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
mogsa <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=9,
               proc.row = "center_ssq1", w.data = "inertia", stasis = TRUE)

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

---

prepGraphite

*Prepare pathway gene sets from graphite package*

---

**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**

Sales G, Calura E and Romualdi C (2014). graphite: GRAPH Interaction from pathway Topological Environment. R package version 1.10.1.

**See Also**

See Also as [prepMsigDB](#) and [prepSupMoa](#).

**Examples**

```
library(graphite)
keggdb <- prepGraphite(db = pathways("hsapiens", "kegg")[1:3], id = "entrez")
```

---

prepMsigDB	<i>Conver gmt format file to a list</i>
------------	---

---

**Description**

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

**Usage**

```
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**

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

---

prepSupMoa

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

---

**Description**

Convert a list of gene set information to a set of supplementary tables that can be used as input of function "sup.moa" or "mogsa".

**Usage**

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

**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 be 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 <a href="#">prepGraphite</a> or <a href="#">prepMolsigDB</a> .
minMatch	The minimum match of geneset.
maxMatch	The maximum match genesets.

**Details**

Details here

**Value**

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

**Author(s)**

Chen Meng

**See Also**

See Also as [prepGraphite](#) and [prepMsigDB](#).

## Examples

```
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)
```

---

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)
```

## Arguments

X	An object of class <a href="#">moa-class</a>
sup	A list of data.frames contains supplementary data.
nf	The number of principal components used in the projection.

## 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, Dominique Valentin and Mohammed Bennani-Dosse. STATIS and DISTATIS: optimum multitable principal component analysis and three way metric multidimensional scaling. WIREs Comput Stat 2012. Volume 4, Issue 2, pages 124-167

**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_ssq1", w.data = "inertia", stasis = TRUE)
plot(ana, value="eig")
# projectin supplementary data
smao <- sup.moa(ana, sup=NCI60_4array_supdata, nf=3)
# heatmap visualize the gene set scores
heatmap(slot(smao, "score"))

```

---

wsvd

*Weighted singular value decomposition (SVD)*


---

**Description**

The weighted version of singular value decomposition.

**Usage**

```
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 D1 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**

Herve Abdi. Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD) <http://www.utdallas.edu/~herve/Abdi-SVD2007-pretty.pdf>

**See Also**

svd

**Examples**

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

# Index

- \*Topic **MFA**
    - moa, [14](#)
  - \*Topic **MVA**
    - moa, [14](#)
    - mogsa, [18](#)
  - \*Topic **Microarray**
    - NCI60\_4array\_supdata, [21](#)
    - NCI60\_4arrays, [20](#)
  - \*Topic **NCI-60**
    - NCI60\_4array\_supdata, [21](#)
    - NCI60\_4arrays, [20](#)
  - \*Topic **PCA**
    - moa, [14](#)
  - \*Topic **RV coefficient**
    - pairwise.rv, [21](#)
  - \*Topic **STATIS**
    - moa, [14](#)
  - \*Topic **SVD**
    - wsvd, [28](#)
  - \*Topic **classes**
    - mgsa-class, [13](#)
    - moa-class, [15](#)
    - moa.sup-class, [17](#)
  - \*Topic **combine**
    - combine-methods, [5](#)
  - \*Topic **data projection**
    - sup.moa, [27](#)
  - \*Topic **datasets**
    - NCI60\_4array\_supdata, [21](#)
    - NCI60\_4arrays, [20](#)
  - \*Topic **generalized SVD**
    - wsvd, [28](#)
  - \*Topic **graphite**
    - prepGraphite, [24](#)
  - \*Topic **methods**
    - plot-methods, [22](#)
  - \*Topic **mgsa-class**
    - combine-methods, [5](#)
  - \*Topic **moa-class**
    - plot-methods, [22](#)
  - \*Topic **mogsa**
    - combine-methods, [5](#)
  - \*Topic **pahtways**
    - prepGraphite, [24](#)
  - \*Topic **supplementary data projection**
    - mogsa, [18](#)
  - \*Topic **supplementary data**
    - sup.moa, [27](#)
  - \*Topic **weighted SVD**
    - wsvd, [28](#)
- annotate.gs, [3](#), [11](#)
- box.gs.feature, [4](#), [18](#)
- boxplot, [4](#)
- combine (combine-methods), [5](#)
- combine, mgsa, mgsa-method (combine-methods), [5](#)
- combine-methods, [5](#)
- decompose.gs.group, [6](#), [8](#), [18](#)
- decompose.gs.ind, [7](#), [7](#), [18](#)
- getmgsa, [9](#)
- GIS, [3](#), [10](#)
- matpower, [12](#)
- mgsa-class, [13](#)
- moa, [4](#), [13](#), [14](#), [18](#), [19](#)
- moa-class, [15](#)
- moa.sup, [13](#)
- moa.sup-class, [17](#)
- mogsa, [15](#), [18](#)
- mogsa-package, [2](#)
- NCI60\_4array\_supdata, [21](#)
- NCI60\_4arrays, [20](#)
- pairwise.rv, [21](#)

plot, moa, missing-method (moa-class), [15](#)  
plot-methods, [22](#)  
plotGS, [18](#), [23](#)  
points, [23](#)  
prepGraphite, [24](#), [26](#)  
prepMsigDB, [25](#), [25](#), [26](#)  
prepSupMoa, [25](#), [26](#), [26](#)

slot, [9](#)  
sup.moa, [15](#), [18](#), [19](#), [27](#)

text, [23](#)

wsvd, [12](#), [28](#)