

# Package ‘made4’

May 29, 2024

**Version** 1.78.0

**Title** Multivariate analysis of microarray data using ADE4

**Author** Aedin Culhane

**Maintainer** Aedin Culhane <Aedin@jimmy.harvard.edu>

**Imports** ade4

**Depends** RColorBrewer,gplots,scatterplot3d, Biobase,  
SummarizedExperiment

**Suggests** affy, BiocStyle, knitr, rmarkdown

**Description** Multivariate data analysis and graphical display of microarray data. Functions include for supervised dimension reduction (between group analysis) and joint dimension reduction of 2 datasets (coinertia analysis). It contains functions that require R package ade4.

**Reference** Culhane AC, Thioulouse J, Perriere G, Higgins DG. (2005) MADE4: an R package for multivariate analysis of gene expression data. *Bioinformatics*. 21(11):2789-90. Culhane AC, Thioulouse J (2006) A multivariate approach to integrating datasets using made4 and ade4. *R News: Special Issue on Bioconductor December*.

**License** Artistic-2.0

**VignetteBuilder** knitr

**LazyData** TRUE

**URL** <http://www.hsph.harvard.edu/aedin-culhane/>

**biocViews** Clustering, Classification, DimensionReduction, PrincipalComponent,Transcriptomics, MultipleComparison, GeneExpression, Sequencing, Microarray

**git\_url** <https://git.bioconductor.org/packages/made4>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 981e55a

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-29

## Contents

asDataFrame	2
bet.coinertia	3
between.graph	5
bga	6
bga.jackknife	9
bga.suppl	10
checkfac	12
chime3D	12
cia	14
commonMap	16
comparelists	18
do3d	19
dudi.rwcoa	21
forrwcoa	22
genes	23
genes1d	23
getcol	25
graph1D	26
heatplot	27
html3D	30
isDataFrame	32
jmol3D	33
khan	34
NCI60	36
ord	38
overview	40
plotarrays	41
plotgenes	43
prettyDend	44
randomiser	46
s.var	47
sumstats	49
suppl	50
topgenes	52
<b>Index</b>	<b>54</b>

---

asDataFrame

*Check that input data is a data.frame*

---

### Description

This function is called by isDataFrame which converts input data into a data.frame suitable for analysis in ADE4. This function is not normally called by the user

**Usage**

```
asDataFrame(indata)
```

**Arguments**

`indata` A [matrix](#), [data.frame](#), [ExpressionSet](#) or [marrayRaw-class](#) or [RangedSummarizedExperiment](#), [SummarizedExperiment](#). If the input is gene expression data in a [matrix](#) or [data.frame](#). The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.

**Details**

[bga](#) and other functions in [made4](#) call this function and it is generally not necessary to call this directly. [isDataFrame](#) calls [isDataFrame](#), and will accept a [matrix](#), [data.frame](#), [ExpressionSet](#) or [marrayRaw-class](#), [SummarizedExperiment](#), [RangedSummarizedExperiment](#) format. It will also transpose data or add a integer to generate a positive data matrix. If the input data contains NA, these must first be removed or imputed (see the R libraries [impute\(\)](#) or [pamr\(\)](#)).

**Value**

Returns a [data.frame](#) suitable for analysis by [ade4](#) or [made4](#) functions.

**Note**

This is not normally called by a user. Internal function.

**Author(s)**

Aedin Culhane

**See Also**

[isDataFrame](#)

---

bet.coinertia

*Between class coinertia analysis*

---

**Description**

Between class coinertia analysis. [cia](#) of 2 datasets where covariance between groups or classes of cases, rather than individual cases are maximised.

**Usage**

```
bet.coinertia(df1, df2, fac1, fac2, cia.nf = 2, type = "nsc", ...)
```

**Arguments**

df1	First dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
df2	Second dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
fac1	A factor or vector which describes the classes in df1.
fac2	A factor or vector which describes the classes in df2.
cia.nf	Integer indicating the number of coinertia analysis axes to be saved. Default value is 2.
type	A character string, accepted options are type="nsc" or type="pca".
...	further arguments passed to or from other methods.

**Value**

A list of class `bet.cia` of length 5

coin	An object of class 'coinertia', sub-class <a href="#">dudi</a> . See <a href="#">coinertia</a>
coa1, pca1	An object of class 'nsc' or 'pca', with sub-class 'dudi'. See <a href="#">dudi</a> , <a href="#">dudi.pca</a> or <a href="#">dudi.nsc</a>
coa2, pca2	An object of class 'nsc' or 'pca', with sub-class 'dudi'. See <a href="#">dudi</a> , <a href="#">dudi.pca</a> or <a href="#">dudi.nsc</a>
bet1	An object of class 'bga', with sub-class 'dudi'. See <a href="#">dudi</a> , <a href="#">bga</a> or <a href="#">bca</a>
bet2	An object of class 'bga', with sub-class 'dudi'. See <a href="#">dudi</a> , <a href="#">bga</a> or <a href="#">bca</a> .

**Note**

This is very computational intensive. The authors of `ade4` are currently re-writing the code for coinertia analysis, so that it should substantially improve the computational requirements (May 2004).

**Author(s)**

Aedin Culhane

**References**

Culhane AC, et al., 2003 Cross platform comparison and visualisation of gene expression data using co-inertia analysis. *BMC Bioinformatics*. 4:59

**See Also**

See Also as [coinertia](#), [cia](#).

**Examples**

```
### NEED TO DO
if (require(ade4, quiet = TRUE)) {}
```

---

between.graph	<i>Plot 1D graph of results from between group analysis</i>
---------------	---

---

**Description**

Plots a 1D graph, of results of between group analysis similar to that in Culhane et al., 2002.

**Usage**

```
between.graph(x, ax = 1, cols = NULL, hor = TRUE, scaled=TRUE,
centnames=NULL, varnames=NULL, ...)
```

**Arguments**

x	Object of the class bga resulting from a <a href="#">bga</a> analysis.
ax	Numeric. The column number of principal component ( $\$s$ and $\$i$ ) to be used. Default is 1. This is the first component of the analysis.
cols	Vector of colours. By default colours are obtained using <a href="#">getcol</a>
hor	Logical, indicating whether the graph should be plotted horizontally or vertically. The default is a horizontal plot.
scaled	Logical, indicating whether the coordinates in the graph should be scaled to fit optimally in plot. Default is TRUE
centnames	A vector of variables labels. Default is NULL, if NULL the row names of the centroid $\$i$ coordinates will be used.
varnames	A vector of variables labels. Default is NULL, if NULL the row names of the variable $\$s$ coordinates will be used.
...	further arguments passed to or from other methods

**Details**

This will produce a figure similar to Figure 1 in the paper by Culhane et al., 2002.

between.graph requires both samples and centroid co-ordinates ( $\$s$ ,  $\$i$ ) which are passed to it via an object of class bga. If cases are to be coloured by class, it also requires a  $\$fac$  factor which is also passed to it via an object of class bga.

To plot a 1D graph from other multivariate analysis such as PCA ([dudi.pca](#)), COA ([dudi.coa](#)), or [coinertia](#) analysis. Please use [graph1D](#).

**Author(s)**

Aedin Culhane

## References

Culhane AC, et al., 2002 Between-group analysis of microarray data. *Bioinformatics*. 18(12):1600-8.

## See Also

[graph1D](#)

## Examples

```
data(khan)
if (require(ade4, quiet = TRUE)) {
khan.bga<-bga(khan$train, khan$train.classes)
}

between.graph(khan.bga)
between.graph(khan.bga, ax=2, lwd=3, cex=0.5, col=c("green","blue", "red", "yellow"))
between.graph(khan.bga, ax=2, hor=FALSE, col=c("green","blue", "red", "yellow"))
```

---

bga

*Between group analysis*

---

## Description

Discrimination of samples using between group analysis as described by Culhane et al., 2002.

## Usage

```
bga(dataset, classvec, type = "coa", ...)
## S3 method for class 'bga'
plot(x, axis1=1, axis2=2, arraycol=NULL, genecol="gray25", nlab=10,
     genelabels= NULL, ...)
```

## Arguments

dataset	Training dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
classvec	A factor or vector which describes the classes in the training dataset.
type	Character, "coa", "pca" or "nsc" indicating which data transformation is required. The default value is type="coa".
x	An object of class bga. The output from bga or <a href="#">bga.suppl</a> . It contains the projection coordinates from bga, the $\$ls$ , $\$co$ or $\$li$ coordinates to be plotted.

arraycol, genecol	Character, colour of points on plot. If arraycol is NULL, arraycol will obtain a set of contrasting colours using <code>getcol</code> , for each classes of cases (microarray samples) on the array (case) plot. <code>genecol</code> is the colour of the points for each variable (genes) on gene plot.
nlab	Numeric. An integer indicating the number of variables (genes) at the end of axes to be labelled, on the gene plot.
axis1	Integer, the column number for the x-axis. The default is 1.
axis2	Integer, the column number for the y-axis, The default is 2.
genelabels	A vector of variables labels, if <code>genelabels=NULL</code> the <code>row.names</code> of input matrix dataset will be used.
...	further arguments passed to or from other methods.

## Details

`bga` performs a between group analysis on the input dataset. This function calls `bca`. The input format of the dataset is verified using `isDataFrame`.

Between group analysis is a supervised method for sample discrimination and class prediction. BGA is carried out by ordinating groups (sets of grouped microarray samples), that is, groups of samples are projected into a reduced dimensional space. This is most easily done using PCA or COA, of the group means. The choice of PCA, COA is defined by the parameter `type`.

The user must define microarray sample groupings in advance. These groupings are defined using the input `classvec`, which is a factor or vector.

### Cross-validation and testing of bga results:

`bga` results should be validated using one leave out jack-knife cross-validation using `bga.jackknife` and by projecting a blind test datasets onto the `bga` axes using `suppl`. `bga` and `suppl` are combined in `bga.suppl` which requires input of both a training and test dataset. It is important to ensure that the selection of cases for a training and test set are not biased, and generally many cross-validations should be performed. The function `randomiser` can be used to randomise the selection of training and test samples.

**Plotting and visualising bga results:** *1D plots, show one axis only:* 1D graphs can be plotted using `between.graph` and `graph1D`. `between.graph` is used for plotting the cases, and required both the co-ordinates of the cases ( $\$s$ ) and their centroids ( $\$li$ ). It accepts an object `bga`. `graph1D` can be used to plot either cases (microarrays) or variables (genes) and only requires a vector of coordinates.

*2D plots:* Use `plot.bga` to plot results from `bga`. `plot.bga` calls the functions `plotarrays` to draw an xy plot of cases ( $\$s$ ). `plotgenes`, is used to draw an xy plot of the variables (genes). `plotgenes`, is used to draw an xy plot of the variables (genes).

*3D plots:* 3D graphs can be generated using `do3D` and `html3D`. `html3D` produces a web page in which a 3D plot can be interactively rotated, zoomed, and in which classes or groups of cases can be easily highlighted.

### Analysis of the distribution of variance among axes:

It is important to know which cases (microarray samples) are discriminated by the axes. The number of axes or principal components from a `bga` will equal the number of classes - 1, that is `length(levels(classvec))-1`.

The distribution of variance among axes is described in the eigenvalues ( $\lambda_{seig}$ ) of the bga analysis. These can be visualised using a scree plot, using `scatterutil.eigen` as it done in `plot.bga`. It is also useful to visualise the principal components from a using a bga or principal components analysis `dudi.pca`, or correspondence analysis `dudi.coa` using a heatmap. In MADE4 the function `heatplot` will plot a heatmap with nicer default colours.

#### Extracting list of top variables (genes):

Use `topgenes` to get list of variables or cases at the ends of axes. It will return a list of the top n variables (by default n=5) at the positive, negative or both ends of an axes. `sumstats` can be used to return the angle (slope) and distance from the origin of a list of coordinates.

For more details see Culhane et al., 2002 and <http://bioinf.ucd.ie/research/BGA>.

#### Value

A list with a class bga containing:

<code>ord</code>	Results of initial ordination. A list of class "dudi" (see <code>dudi</code> )
<code>bet</code>	Results of between group analysis. A list of class "dudi" (see <code>dudi</code> ), "between" (see <code>bca</code> )
<code>fac</code>	The input classvec, the factor or vector which described the classes in the input dataset

#### Author(s)

Aedin Culhane

#### References

Culhane AC, et al., 2002 Between-group analysis of microarray data. *Bioinformatics*. 18(12):1600-8.

#### See Also

See Also `bga`, `suppl`, `suppl.bga`, `bca`, `bga.jackknife`

#### Examples

```
data(khan)

if (require(ade4, quiet = TRUE)) {
  khan.bga<-bga(khan$train, classvec=khan$train.classes)
}

khan.bga
plot(khan.bga, genelabels=khan$annotation$Symbol)

# Provide a view of the principal components (axes) of the bga
heatplot(khan.bga$bet$ls, dend="none")
```



---

bga.jackknife      *Jackknife between group analysis*

---

### Description

Performs one-leave-out jackknife analysis of a between group analysis as described by Culhane et al., 2002

### Usage

```
bga.jackknife(data, classvec, ...)
```

### Arguments

data	Input dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The columns contain the cases (array samples) which will be jackknifed.
classvec	A factor or vector which describes the classes in the training dataset
...	further arguments passed to or from other methods

### Details

Performs a one-leave-out cross validation of between group analysis [bga](#). Input is a training dataset. This can take 5-10 minutes to compute on standard data gene expression matrix.

In jackknife one leave out analysis, one case (column) is removed. The remaining dataset is subjected to [bga](#). Then the class of the case that was removed is predicted using [suppl](#). This analysis is repeated until all samples have been removed and predicted.

### Value

A list containing

results	The projected co-ordinates of each sample
summary	A summary of number and percentage of correctly assigned samples

### Author(s)

Aedin Culhane

### References

Culhane et al., 2002 Between-group analysis of microarray data. *Bioinformatics*. 18(12):1600-8.

### See Also

See Also [bga](#), [bga.suppl](#), [suppl.bga](#), [bca](#), [plot.bga](#)

**Examples**

```

data(khan)
# NOTE using a very reduced dataset (first 5 genes) to speed up results
# hence expect poor prediction accuracy
dim(khan$train)
print("using only small subset of data")
if (require(ade4, quiet = TRUE)) {
  bga.jackknife(khan$train[1:5,], khan$train.classes) }

```

bga.suppl

*Between group analysis with supplementary data projection***Description**

bga.suppl performs a bga between group analysis with projection of supplementary points using suppl

**Usage**

```
bga.suppl(dataset, supdata, classvec, supvec = NULL, suponly = FALSE, type="coa", ...)
```

**Arguments**

dataset	Training dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
supdata	Test or blind dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively. The test dataset supdata and the training dataset dataset must contain the same number of variables (genes).
classvec	A factor or vector which describes the classes in the training data dataset.
supvec	A factor or vector which describes the classes in the test dataset supdata.
suponly	Logical indicating whether the returned output should contain the test class assignment results only. The default value is FALSE, that is the training coordinates, test coordinates and class assignments will all be returned.
type	Character, "coa", "pca" or "nsc" indicating which data transformation is required. The default value is type="coa".
...	further arguments passed to or from other methods.

**Details**

`bga.suppl` calls `bga` to perform between group analysis (`bga`) on the training dataset, then it calls `suppl` to project the test dataset onto the `bga` axes. It returns the coordinates and class assignment of the cases (microarray samples) in the test dataset as described by Culhane et al., 2002.

The test dataset must contain the same number of variables (genes) as the training dataset.

The input format of both the training dataset and test dataset are verified using `isDataFrame`. Use `plot.bga` to plot results from `bga`.

**Value**

If `suponly` is `FALSE` (the default option) `bga.suppl` returns a list of length 4 containing the results of the `bga` of the training dataset and the results of the projection of the test dataset onto the `bga` axes-

<code>ord</code>	Results of initial ordination. A list of class "dudi" (see <a href="#">dudi</a> ).
<code>bet</code>	Results of between group analysis. A list of class "dudi" (see <a href="#">dudi</a> ), "between" (see <a href="#">bca</a> ), and "dudi.bga" (see <a href="#">bga</a> )
<code>fac</code>	The input <code>classvec</code> , the factor or vector which described the classes in the input dataset
<code>suppl</code>	An object returned by <a href="#">suppl</a>

If `suponly` is `TRUE` only the results from `suppl` will be returned.

**Author(s)**

Aedin Culhane

**References**

Culhane AC, et al., 2002 Between-group analysis of microarray data. *Bioinformatics*. 18(12):1600-8.

**See Also**

See Also [bga](#), [suppl](#), [bca](#), [plot.bga](#), [bga.jackknife](#)

**Examples**

```
data(khan)
#khan.bga<-bga(khan$train, khan$train.classes)
if (require(ade4, quiet = TRUE)) {
khan.bga<-bga.suppl(khan$train, supdata=khan$test,
classvec=khan$train.classes, supvec=khan$test.classes)

khan.bga
plot.bga(khan.bga, genelabels=khan$annotation$Symbol)
khan.bga$suppl
}
```

---

checkfac	<i>Check factor</i>
----------	---------------------

---

**Description**

Convert input [vector](#) into a [factor](#) suitable for analysis in ADE4. This function is called by `link[made4:bga]{bga}` and other made4 function, it is not normally called by the user

**Usage**

```
checkfac(classvec)
```

**Arguments**

classvec	A <a href="#">vector</a> or <a href="#">factor</a>
----------	--

**Value**

factor	<a href="#">factor</a> suitable for input into bga
--------	--

**Author(s)**

Aedin Culhane

---

chime3D	<i>Produce web graph of 3D graph that can be viewed using Chime web browser plug-in from a pdb file</i>
---------	---

---

**Description**

chime3D produces a html web page with a 3D graph of a pdb file that can be rotated and manipulated in a web browser that supports the chime plug-in.

**Usage**

```
chime3D(pdbfilename, classvec = NULL, title = NULL, filename = "output.html",  
point.size=40,cols=NULL, ...)
```

**Arguments**

pdbfilename	Input pdb filename, this is an output file from htm13D.
classvec	Factor or vector which describes classes in the df. Default is NULL. If specified each group will be coloured in contrasting colours
title	Character, the title (header) of the web page saved if writehtml is TRUE. The default is NULL.
filename	Character. The filename of the html to be saved. The default is "output.html".
point.size	Numeric, size of the points in plot. Default is 40
cols	Vector, of colours to be used. The length of the vector must equal the length levels(classvec), that is the number of classes or groups of samples. The default is NULL, colours will be chosen using getcol.
...	further arguments passed to or from other methods

**Details**

Produces a html file, of a 3D graph which can be rotated using the FREEWARE chime (win, MacOS). Will colour samples by classvec if given one, and will produce chime script to highlight groups, spin on/off, and include button for restore for example see <http://bioinf.ucd.ie/research/microarrays/>.

**Value**

chime3D output is a html file that can be viewed in any web browser that supports the Chime plug-in.

**Note**

Note chime is only available on windows or Mac OS currently. Using the chime plug-in on Linux is slightly complicated but is available if the CrossOver Plug-in is installed. Instructions on installing this and chime on Linux are available at [http://mirrors.rcsb.org/SMS/STINGm/help/chime\\_linux.html](http://mirrors.rcsb.org/SMS/STINGm/help/chime_linux.html).

If you wish to view a 3D graph in Rasmol, you will need to execute a Rasmol script similar to

```
load pdbfilename.pdb;
set axes on;
select off;
connect;
set ambient 40;
rotate x 180;
select *;
spacefill 40
```

**Author(s)**

Aedin Culhane

**Description**

Performs CIA on two datasets as described by Culhane et al., 2003. Used for meta-analysis of two or more datasets.

**Usage**

```
cia(df1, df2, cia.nf=2, cia.scan=FALSE, nsc=TRUE,...)
## S3 method for class 'cia'
plot(x, nlab = 10, axis1 = 1, axis2 = 2, genecol = "gray25",
      genelabels1 = rownames(ciares$co), genelabels2 = rownames(ciares$li), ...)
```

**Arguments**

df1	The first dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
df2	The second dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
cia.nf	Integer indicating the number of coinertia analysis axes to be saved. Default value is 2.
cia.scan	Logical indicating whether the coinertia analysis eigenvalue (scree) plot should be shown so that the number of axes, <code>cia.nf</code> can be selected interactively. Default value is FALSE.
nsc	A logical indicating whether coinertia analysis should be performed using two non-symmetric correspondence analyses <a href="#">dudi.nsc</a> . The default=TRUE is highly recommended. If FALSE, COA <a href="#">dudi.coa</a> will be performed on df1, and row weighted COA <a href="#">dudi.rwcoa</a> will be performed on df2 using the row weights from df1.
x	An object of class <code>cia</code> , containing the CIA projected coordinates to be plotted.
nlab	Numeric. An integer indicating the number of variables (genes) to be labelled on plots.
axis1	Integer, the column number for the x-axis. The default is 1.
axis2	Integer, the column number for the y-axis. The default is 2.
genecol	Character, the colour of genes (variables). The default is "gray25".
genelabels1, genelabels2	A vector of variables labels, by default the row.names of each input matrix df1, and df2 are used.
...	further arguments passed to or from other methods.

## Details

CIA has been successfully applied to the cross-platform comparison (meta-analysis) of microarray gene expression datasets (Culhane et al., 2003). Please refer to this paper and the vignette for help in interpretation of the output from CIA.

Co-inertia analysis (CIA) is a multivariate method that identifies trends or co-relationships in multiple datasets which contain the same samples. That is the rows or columns of the matrix have to be weighted similarly and thus must be "matchable". In `cia`, it is assumed that the analysis is being performed on the microarray cases, and thus the columns will be matched between the 2 datasets. Thus please ensure that the order of cases (the columns) in `df1` and `df2` are equivalent before performing CIA.

CIA simultaneously finds ordinations (dimension reduction diagrams) from the datasets that are most similar. It does this by finding successive axes from the two datasets with maximum covariance. CIA can be applied to datasets where the number of variables (genes) far exceeds the number of samples (arrays) such is the case with microarray analyses.

`cia` calls `coinertia` in the ADE4 package. For more information on coinertia analysis please refer to `coinertia` and several recent reviews (see below).

In the paper by Culhane et al., 2003, the datasets `df1` and `df2` are transformed using COA and Row weighted COA respectively, before coinertia analysis. It is now recommended to perform non symmetric correspondence analysis (NSC) rather than correspondence analysis (COA) on both datasets.

### The RV coefficient

In the results, in the object `cia` returned by the analysis, `\$coinertia\$RV` gives the RV coefficient. This is a measure of global similarity between the datasets, and is a number between 0 and 1. The closer it is to 1 the greater the global similarity between the two datasets.

### Plotting and visualising cia results

`plot.cia` draws 3 plots.

The first plot uses `S.match.col` to plots the projection (normalised scores  $\$mY$  and  $\$mX$ ) of the samples from each dataset onto the one space. Cases (microarray samples) from one dataset are represented by circles, and cases from the second dataset are represented by arrow tips. Each circle and arrow is joined by a line, where the length of the line is proportional to the divergence between the gene expression profiles of that sample in the two datasets. A short line shows good agreement between the two datasets.

The second two plots call `plot.genes` are show the projection of the variables (genes,  $\$li$  and  $\$co$ ) from each dataset in the new space. It is important to note both the direction of project of Variables (genes) and cases (microarray samples). Variables and cases that are projected in the same direction from the origin have a positive correlation (ie those genes are upregulated in those microarray samples)

*Please refer to the help on `bga` for further discussion on graphing and visualisation functions in MADE4.*

## Value

An object of the class `cia` which contains a list of length 4.

`call`                    list of input arguments, `df1` and `df2`

coinertia	A object of class "coinertia", sub-class dudi. See <a href="#">coinertia</a>
coa1	Returns an object of class "coa" or "nsc", with sub-class <a href="#">dudi</a> . See <a href="#">dudi.coa</a> or <a href="#">dudi.nsc</a>
coa2	Returns an object of class "coa" or "nsc", with sub-class <a href="#">dudi</a> . See <a href="#">dudi.coa</a> or <a href="#">dudi.nsc</a>

**Author(s)**

Aedin Culhane

**References**

Culhane AC, et al., 2003 Cross platform comparison and visualisation of gene expression data using co-inertia analysis. BMC Bioinformatics. 4:59

**See Also**

See also [coinertia](#), [plot.cia](#). CIA and multiple CIA is also implemented in Bioconductor packages [omicade4](#) and [mogsa](#)

**Examples**

```
data(NCI60)
print("This will take a few minutes, please wait...")

if (require(ade4, quiet = TRUE)) {
# Example data are "G1_Ross_1375.txt" and "G5_Affy_1517.txt"
coin <- cia(NCI60$Ross, NCI60$Affy)
}
attach(coin)
summary(coin)
summary(coin$coinertia)
# $coinertia$RV will give the RV-coefficient, the greater (scale 0-1) the better
cat(paste("The RV coefficient is a measure of global similarity between the datasets.\n",
"The two datasets analysed are very similar. ",
"The RV coefficient of this coinertia analysis is: ", coin$coinertia$RV, "\n", sep= ""))
plot(coin)
plot(coin, classvec=NCI60$classvec[,2], clab=0, cpoint=3)
```

---

commonMap

*Plot highlights common points between two 1D plots (bipartite)*

---

**Description**

CommonMap draws two 1D plots, and links the common points between the two.

**Usage**

```
commonMap(x, y, hor=TRUE, cex=1.5, scaled=TRUE, ...)
```



**Arguments**

x	The coordinates of the first axis
y	The coordinates of the second axis
hor	Logical, whether a horizontal line should be drawn on plot. Default is TRUE.
cex	Numeric. The amount by which plotting text and symbols should be scaled relative to the default
scaled	Logical, whether the data in x and y are scaled. Scaling is useful for visualising small or large data values. Set to FALSE if actually or true values should be visualised. The default is TRUE.
...	further arguments passed to or from other method

**Details**

Useful for mapping the genes in common from coinertia analysis This graphs a 1D graph, x and y are the coordinates from two different analyses but the rows of each vectors correspond (ie common genes)

**Note**

This is useful for examining common points in axes from coinertia analysis, or comparing results from two different analysis.

**Author(s)**

Ailis Fagan and Aedin Culhane

**See Also**

See also [between.graph](#), [graph1D](#)

**Examples**

```
a<-rnorm(20)
b<-rnorm(20)
par(mfrow=c(2,2))
commonMap(a,b)
commonMap(a,b,hor=FALSE, col="red", pch=19)
commonMap(a,b,col="blue", cex=2, pch=19)

# If the vectors contain different variables, the rows should define the variables that correspond
a[15:20]<-NA
b[10:15]<-NA
cbind(a,b)
commonMap(a,b, col="dark green", pch=18)
```

---

 comparelists

*Return the intersect, difference and union between 2 vectors*


---

### Description

This is a very simple function which compares two vectors, x and y. It returns the intersection and unique lists. It is useful for comparing two genelists.

### Usage

```
comparelists(dx,dy, ...)
## S3 method for class 'comparelists'
print(x, ...)
```

### Arguments

dx, dy	A vector.
x	An object from comparelists.
...	further arguments passed to or from other methods.

### Details

reports on the intersect, difference and union between two lists.

### Value

An object of class comparelists:

intersect	Vector containing the intersect between x and y
Set.Diff	Vector containing the elements unique to X obtained using <a href="#">setdiff</a>
XinY	Numeric, indicating the number of elements of x in y
YinX	Numeric, indicating the number of elements of y in x
Length.X	Numeric, the number of elements in x
Length.Y	Numeric, the number of elements in y
...	Further arguments passed to or from other methods

### Author(s)

Aedin Culhane

### See Also

See also [intersect](#), [setdiff](#)

## Examples

```
a<-sample(LETTERS,20)
b<-sample(LETTERS,10)
z<-comparelists(a,b)
z$Set.Diff
z$intersect
```

---

do3d

*Generate 3D graph(s) using scatterplot3d*


---

## Description

do3d is a wrapper for scatterplot3d. do3d will draw a single 3D xyz plot and will plot each group of points in a different colour, given a factor.

rotate3d calls do3d to draw multiple 3D plots in which each plot is marginally rotated on the x-y axis.

## Usage

```
do3d(dataset, x = 1, y = 2, z = 3, angle = 40, classvec = NULL, classcol
= NULL, col = NULL, cex.lab=0.3, pch=19, cex.symbols=1, ...)
rotate3d(dataset, x = 1, y = 2, z = 3, beg = 180, end = 360, step = 12,
savefiles = FALSE, classvec = NULL, classcol = NULL, col = NULL, ...)
```

## Arguments

dataset	XYZ coordinates to be plotted. A <a href="#">matrix</a> or <a href="#">data.frame</a> with 3 or more columns. Usually results from multivariate analysis, such as the $\$co$ or $\$li$ coordinates from a PCA <a href="#">dudi.pca</a> , or COA <a href="#">dudi.coa</a> or the $\$ls$ , $\$co$ coordinates from <a href="#">bga</a> .
x	Numeric, the column number for the x-axis, the default is 1 (that is dataset[,1])
y	Numeric, the column number for the y-axis, the default is 2 (that is dataset[,2])
z	Numeric, the column number for the z-axis, the default is 3 (that is dataset[,3])
angle	Numeric, the angle between x and y axis. Note the result depends on scaling. See <a href="#">scatterplot3d</a>
classvec	A factor or vector which describes the classes in dataset
classcol	A factor or vector which list the colours for each of the classes in the dataset. By default NULL. When NULL, getcol is used to obtain an optimum set of colours of the classes in classvec.
cex.lab	Numeric. The magnification to be used for the axis annotation relative to the current default text and symbol size. Default is 0.3
pch	Integer specifying a symbol or single character to be used when plotting points. The default is pch= 19

<code>cex.symbols</code>	Numeric. The magnification to be used for the symbols relative to the current default text size. Default is 1
<code>col</code>	A character indicating a colour. To be used if all points are to be one colour. If <code>classvec</code> , <code>classcol</code> and <code>col</code> are all NULL. all points will be drawn in red by default.
<code>beg</code>	Numeric. The starting angle between the x and y axis for <code>rotate3d</code> . <code>Rotate3d</code> will draw plots in which they are rotated from angle <code>beg</code> to angle <code>end</code>
<code>end</code>	Numeric. The final angle between the x and y axis for <code>rotate3d</code> . <code>Rotate3d</code> will draw plots in which they are rotated from angle <code>beg</code> to angle <code>end</code>
<code>step</code>	Numeric. Increment of the sequence between the starting angle <code>beg</code> and the final angle <code>end</code> .
<code>savefiles</code>	Logical, indicating whether the plot should be saved as a pdf file. The default is FALSE
<code>...</code>	further arguments passed to or from other methods

### Details

This calls `scatterplot3d` to plot a 3d representation of results.

It is also worth exploring the package `rgl` which enables dynamic 3d plot (that can be rotated)

```
library(rgl) plot3d(khan.coa$co[,1], khan.coa$co[,2],khan.coa$co[,3], size=4, col=khan$train.classes)
rgl.snapshot(file="test.png", top=TRUE) rgl.close()
```

### Value

Produces plots of the xyz coordinates.

### Author(s)

Aedin Culhane

### See Also

See Also [scatterplot3d](#)

### Examples

```
data(khan)
if (require(ade4, quiet = TRUE)) {
khan.coa<-dudi.coa(khan$train, scannf=FALSE, nf=5)
}
par(mfrow=c(2,1))
do3d(khan.coa$co, classvec=khan$train.classes)
do3d(khan.coa$co, col="blue")
rotate3d(khan.coa$co,classvec=khan$train.classes)
khan.bga<-bga(khan$train, khan$train.classes)
plot.new()
par(bg="black")
do3d(khan.bga$bet$ls, classvec=khan$train.classes)
```

---

dudi.rwcoa	<i>Row weighted Correspondence Analysis</i>
------------	---

---

**Description**

dudi.rwcoa Row weighted COA, calls forrwcoa to perform row weighted correspondence analysis.

**Usage**

```
dudi.rwcoa(df, rowweights = rep(1/nrow(df),nrow(df)), ...)
```

**Arguments**

df	a data.frame containing positive or null values. It should not contain missing (NA) values.
rowweights	a vector of row weights (by default, uniform row weights)
...	further arguments passed to or from other methods )

**Details**

Performs row weighted COA. Calls forrwcoa to calculates weights.

**Value**

Returns a list of class 'coa', 'rwcoa', and 'dudi' (see [dudi](#))

**Note**

In the paper by Culhane et al., 2002, coinertia analysis was performed with two COAs, a standard [COA](#) and a row weighted COA dudi.rwcoa, on the two gene expression datasets. However it is now recommended to perform two non-symmetric COA, instead of two COA. This avoids having to force the row weights from one analysis on the second. To perform non-symmetric correspondence coinertia analysis, use [bet.coinertia](#).

**Author(s)**

Aedin Culhane, A.B. Dufour

**References**

Culhane AC, et al., 2003 Cross platform comparison and visualisation of gene expression data using co-inertia analysis. BMC Bioinformatics. 4:59

**See Also**

See Also as [dudi](#),[dudi.coa](#),[dudi.pca](#)

---

`forrwcoa`*Row weighted Correspondence Analysis*

---

**Description**

`dudi.rwcoa` Row weighted COA, calls `forrwcoa` to perform row weighted correspondence analysis.

**Usage**

```
forrwcoa(df, rowweights = rep(1/nrow(df),nrow(df)))
```

**Arguments**

<code>df</code>	a data.frame containing positive or null values. It should not contain missing (NA) values.
<code>rowweights</code>	a vector of row weights (by default, uniform row weights)
<code>...</code>	further arguments passed to or from other methods )

**Details**

Performs row weighted COA. Calls `forrwcoa` to calculates weights.

**Value**

Returns a list of class `'coa'`, `'rwcoa'`, and `'dudi'` (see [dudi](#))

**Note**

In the paper by Culhane et al., 2002, coinertia analysis was performed with two COAs, a standard COA and a row weighted COA `dudi.rwcoa`, on the two gene expression datasets. However it is now recommended to perform two non-symmetric COA, instead of two COA. This avoids having to force the row weights from one analysis on the second. To perform non-symmetric correspondence coinertia analysis, use [bet.coinertia](#).

**Author(s)**

Aedin Culhane, A.B. Dufour

**References**

Culhane AC, et al., 2003 Cross platform comparison and visualisation of gene expression data using co-inertia analysis. BMC Bioinformatics. 4:59

**See Also**

See Also as [dudi](#),[dudi.coa](#),[dudi.pca](#) [bet.coinertia](#)

---

genes	<i>Returns an index of the top N variables at the ends of the x and y axes</i>
-------	--

---

**Description**

Returns an index of the top n maximum and minimum values in 2 columns of a matrix or data.frame.

**Usage**

```
genes(dudivar,n=5, axis1=1, axis2=2)
```

**Arguments**

dudivar	a data frame or matrix with at least two columns, containing the x,y coordinates.
axis1	the column number for the x-axis.
axis2	the column number for the y-axis.
n	n is the number of genes to be extracted.

**Details**

Genes returns an index of the "top" variables at the ends of the x and y axes. Genes is not normally called by the user but is called by plotgenes()

If you wish to return a table or list of the top genes at the end of an axis, use the function [topgenes](#).

**Author(s)**

Aedin Culhane

**See Also**

See Also as [plotgenes](#)

---

genes1d	<i>genes1d: list top N variables on one axis. Lists the maximum and minimum values in a column of a data.frame</i>
---------	--

---

**Description**

Lists the top N variables from the positive and negative ends of an axis. Returns a list of N variables that have the max and min. Given a `\$co` or `\$li` file it will return that variables at the ends of the axis.

**Usage**

```
genes1d(array, n = 5, axis = 1, listgenes = FALSE)
```

**Arguments**

array	A <a href="#">matrix</a> or <a href="#">data.frame</a> . Typically a <code>\\$co</code> or <code>\\$li</code> from <a href="#">bga</a> , <a href="#">cia</a> or <a href="#">dudi</a> .
n	An integer indicating the number of variables to be returned. Default is 5.
axis	An integer indicating the column of x. Default is 1 (first axis, of <code>\\$co</code> or <code>\\$li</code> file)
listgenes	A logical, If TRUE a list containing the variables at each ends of the axes are returned
...	further arguments passed to or from other methods

**Details**

Genes1d is similar to `link[made4:genes]{genes}`, but returns an index of genes at the ends of one axes.

**Value**

Returns a vector or list of vectors.

**Author(s)**

Aedin Culhane

**See Also**

See Also as [genes](#)

**Examples**

```
data(khan)
if (require(ade4, quiet = TRUE)) {
khan.coa<-ord(khan$train)
}
ind<-topgenes(khan.coa, ends="pos")
ind.ID<-topgenes(khan.coa, ends="pos", labels=khan$gene.labels.imagesID)
ind.symbol<-topgenes(khan.coa, ends="pos", labels=khan$annotation$Symbol)
Top10.pos<- cbind("Gene Symbol"=ind.symbol,
"Clone ID"=ind.ID, "Coordinates"=khan.coa$ord$co[ind,], row.names=c(1:length(ind)))
Top10.pos
```



---

getcol *Specialised colour palette with set of 21 maximally contrasting colours*

---

### Description

Special colour palette developed to maximise the contrast between colours. Colours were selected for visualising groups of points on xy or xyz plots on a white background. Because of this, there are few pastel colours in this palette. getcol contains 2 palettes of 12 and 21 colours.

### Usage

```
getcol(nc = c(1:3), palette = NULL, test = FALSE)
```

### Arguments

nc	Numeric. Integer or vector in range 1 to 21. This selects colours from palette
palette	A character to select either palette "colours1" or "colours2". colours1 contains 12 colours, colours2 contains 21 colours
test	A logical, if TRUE a plot will be drawn to display the palettes colours1, colours2 and any selected colours.

### Details

Colours1 contains the 12 colours, "red", "blue", "green", "cyan", "magenta", "yellow", "grey", "black", "brown", "orange", "violet", "purple"). These were chosen, as these are compatible with rasmol and chime, that are used in html3D. Colours2 contains 21 colours. These were selected so as to maximise the contrast between groups.

For other colour palettes in R, see colors, palette, rainbow, heat.colors, terrain.colors, topo.colors or cm.colors.

Also see the library RColorBrewer

### Value

A vector containing a list of colours.

### Author(s)

Aedin Culhane

### See Also

See also colors, palette, rainbow, heat.colors, terrain.colors, topo.colors or cm.colors, RColorBrewer

**Examples**

```

getcol(3)
getcol(c(1:7))
getcol(10, test=TRUE)
getcol(c(1:5, 7, 15, 16), palette="colours2", test=TRUE)

```

graph1D

*Plot 1D graph of axis from multivariate analysis***Description**

Draw 1D plot of an axis from multivariate analysis. Useful for visualising an individual axis from analyses such as PCA `dudi.pca` or COA `dudi.coa`. It accepts a factor so that groups of points can be coloured. It can also be used for graphing genes, and will only label `n` genes at the ends of the axis.

**Usage**

```

graph1D(dfx, classvec=NULL, ax = 1, hor=FALSE, s.nam=row.names(dfx), n=NULL,
        scaled=TRUE, col="red", width=NULL, ...)

```

**Arguments**

<code>dfx</code>	<code>vector</code> , <code>matrix</code> , or <code>data.frame</code> , which contains a column with axis coordinates
<code>ax</code>	Numeric, indicating column of <code>matrix</code> , or <code>data.frame</code> to be plotted. The default is 1.
<code>classvec</code>	Factor, indicating sub-groupings or classes in <code>dfx</code> or <code>dfx[,ax]</code>
<code>hor</code>	Logical, indicating whether the graph should be drawn horizontal or vertically. The default is vertically.
<code>s.nam</code>	Vector. labels of <code>dfx</code> , The default is <code>row.names(dfx)</code>
<code>n</code>	Numeric. Whether all rows should be plotted, <code>n=10</code> would label only the 10 variables at the end of the axis. By default all variables (row of <code>dfx</code> ) are labelled
<code>scaled</code>	A logical indicating whether the plot should be scaled to fit. The default is TRUE
<code>col</code>	A character or vector indicating the colour(s) for points or groups of points. If points are to be coloured according to a factor, <code>length(col)</code> should equal <code>length(levels(classvec))</code>
<code>width</code>	A vector of length 2, which is the width (of a vertical plot) or height (of a horizontal plot). This can be increased if variable labels are unreadable. The default is <code>c(-2,1)</code>
<code>...</code>	further arguments passed to or from other methods

**Author(s)**

Aedin Culhane

**See Also**

between.graph

**Examples**

```
a<-rnorm(25)
graph1D(a, s.nam=letters[1:25])
graph1D(a, s.nam=letters[1:25], col="blue", pch=19, n=3)
data(khan)
if (require(ade4, quiet = TRUE)) {
khan.coa<-dudi.coa(khan$train, scan=FALSE, nf=2)
}
graph1D(khan.coa$co, ax=1)
```

---

heatplot

*Draws heatmap with dendrograms.*

---

**Description**

heatplot calls heatmap.2 using a red-green colour scheme by default. It also draws dendrograms of the cases and variables using correlation similarity metric and average linkage clustering as described by Eisen. heatplot is useful for a quick overview or exploratory analysis of data

**Usage**

```
heatplot(dataset, dend = c("both", "row", "column", "none"),
  cols.default = TRUE, lowcol = "green", highcol = "red", scale="none",
  classvec=NULL, classvecCol=NULL, classvec2=NULL, distfun=NULL,
  returnSampleTree=FALSE, method="ave", dualScale=TRUE, zlim=c(-3,3),
  scaleKey=TRUE, ...)
```

**Arguments**

dataset	a <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
dend	A character indicating whether dendrograms should be drawn for both rows and columns "both", just rows "row" or column "column" or no dendrogram "none". Default is both.
cols.default	Logical. Default is TRUE. Use blue-brown color scheme.
lowcol, highcol	Character indicating colours to be used for down and upregulated genes when drawing heatmap if the default colors are not used, that is cols.default = FALSE.
scale	Default is row. Scale and center either "none", "row", or "column").

<code>classvec, classvec2</code>	A factor or vector which describes the classes in columns or rows of the dataset. Default is NULL. If included, a color bar including the class of each column (array sample) or row (gene) will be drawn. It will automatically add to either the columns or row, depending if the <code>length(as.character(classvec)) == nrow(dataset)</code> or <code>ncol(dataset)</code> .
<code>classvecCol</code>	A vector of length the number of levels in the factor <code>classvec</code> . These are the colors to be used for the row or column colorbar. Colors should be in the same order, as the <code>levels(factor(classvec))</code>
<code>distfun</code>	A character, indicating function used to compute the distance between both rows and columns. Defaults to 1- Pearson Correlation coefficient
<code>method</code>	The agglomeration method to be used. This should be one of <code>"ward"</code> , <code>"single"</code> , <code>"complete"</code> , <code>"average"</code> , <code>"mcquitty"</code> , <code>"median"</code> or <code>"centroid"</code> . See <code>hclust</code> for more details. Default is <code>"ave"</code>
<code>dualScale</code>	A logical indicating whether to scale the data by row and columns. Default is TRUE
<code>zlim</code>	A vector of length 2, with lower and upper limits using for scaling data. Default is <code>c(-3,3)</code>
<code>scaleKey</code>	A logical indicating whether to draw a heatmap color-key bar. Default is TRUE
<code>returnSampleTree</code>	A logical indicating whether to return the sample (column) tree. If TRUE it will return an object of class <code>dendrogram</code> . Default is FALSE
.	
...	further arguments passed to or from other methods.

### Details

The hierarchical plot is produced using average linkage cluster analysis with a correlation metric distance. `heatmap` calls [heatmap.2](#) in the R package `gplots`.

NOTE: We have changed `heatmap` scaling in `made4` (v 1.19.1) in Bioconductor v2.5. `Heatplot` by default dual scales the data to limits of -3,3. To reproduce older version of `heatmap`, use the parameters `dualScale=FALSE`, `scale="row"`.

### Value

Plots a heatmap with dendrogram of hierarchical cluster analysis. If `returnSampleTree` is TRUE, it returns an object `dendrogram` which can be manipulated using

### Note

Because Eisen et al., 1998 use green-red colours for the heatmap `heatmap` uses these by default however a blue-red or yellow-blue are easily obtained by changing `lowcol` and `highcol`

### Author(s)

Aedin Culhane

## References

Eisen MB, Spellman PT, Brown PO and Botstein D. (1998). Cluster Analysis and Display of Genome-Wide Expression Patterns. Proc Natl Acad Sci USA 95, 14863-8.

## See Also

See also as [hclust](#), [heatmap](#) and [dendrogram](#)

## Examples

```
data(khan)

## Change color scheme
heatplot(khan$strain[1:30,])
heatplot(khan$strain[1:30,], cols.default=FALSE, lowcol="white", highcol="red")

## Add labels to rows, columns
heatplot(khan$strain[1:26,], labCol = c(64:1), labRow=LETTERS[1:26])

## Add a color bar
heatplot(khan$strain[1:26,], classvec=khan$strain.classes)
heatplot(khan$strain[1:26,], classvec=khan$strain.classes,
classvecCol=c("magenta", "yellow", "cyan", "orange"))

## Change the scaling to the older made4 version (pre Bioconductor 2.5)
heatplot(khan$strain[1:26,], classvec=khan$strain.classes,
dualScale=FALSE, scale="row")

## Getting the members of a cluster and manipulating the tree
sTree<-heatplot(khan$strain, classvec=khan$strain.classes,
returnSampleTree=TRUE)
class(sTree)
plot(sTree)

## Cut the tree at the height=1.0
lapply(cut(sTree,h=1)$lower, labels)

## Zoom in on the first cluster
plot(cut(sTree,1)$lower[[1]])
str(cut(sTree,1.0)$lower[[1]])

## Visualizing results from an ordination using heatplot
if (require(ade4, quiet = TRUE)) {
  # save 5 components from correspondence analysis
  res<-ord(khan$strain, ord.nf=5)
  khan.coa = res$ord
}

# Provides a view of the components of the Correspondence analysis
```

```

# (gene projection)
# first 5 components, do not cluster columns, only rows.
heatmap(khan.coa$li, dend="row", dualScale=FALSE)

# Provides a view of the components of the Correspondence analysis
# (sample projection)
# The difference between tissues and cell line samples
# are defined in the first axis.
# Change the margin size. The default is c(5,5)
heatmap(khan.coa$co, margins=c(4,20), dend="row")

# Add a colorbar, change the heatmap color scheme and no scaling of data
heatmap(khan.coa$co, classvec2=khan$train.classes, cols.default=FALSE,
lowcol="blue", dend="row", dualScale=FALSE)
apply(khan.coa$co, 2, range)

```

---

html3D	<i>Produce web page with a 3D graph that can be viewed using Chime web browser plug-in, and/or a pdb file that can be viewed using Rasmol</i>
--------	---

---

## Description

html3D produces a pdb file that can be viewed using the freeware protein structure viewer Rasmol and a html web page with a 3D graph that can be rotated and manipulated in a web browser that supports the chime web browser plug-in.

## Usage

```
html3D(df, classvec = NULL, writepdb = FALSE, filenamebase = "output",
writehtml = FALSE, title = NULL, scaled=TRUE, xyz.axes=c(1:3), ...)
```

## Arguments

df	A <b>matrix</b> or <b>data.frame</b> containing the x,y,z coordinates. Typically the output from <b>bga</b> such as the <code>\\$ls</code> or <code>\\$co</code> files, or other xyz coordinates ( <code>\\$li</code> or <code>\\$co</code> ) produced by PCA, COA or other <b>dudi</b>
classvec	factor or vector which describes classes in the df. Default is NULL. If specified each group will be coloured in contrasting colours
writepdb	Logical. The default is FALSE. If TRUE a file will be saved which can be read into Rasmol.
writehtml	Logical. The default is FALSE, If TRUE a web html file will be saved which can be viewed in any web browser than supports chime.

filenamebase	Character. The basename of the html or pdb file(s) to be saved. The default is "output", which will save files output.pdb, output.html, if writedb or writehtml are TRUE respectively.
title	Character, the title (header) of the web page saved if writehtml is TRUE. The default is NULL.
scaled	Logical indicating whether the data should be scaled for best fit. The default is TRUE
xyz.axes	vector indicating which axes to use for x, y and z axes. By default, the first 3 columns of df.
...	further arguments passed to or from other methods

### Details

Produces a html file, of a 3D graph which can be rotated using the FREEWARE chime (win, MacOS). Chime can be downloaded from <http://www.mdlchime.com/>.

html3D will colour samples by classvec if given one, and will produce chime script to highlight groups, spin on/off, and include button for restore for example see <http://bioinf.ucd.ie/research/BGA/supplement.html>

html3d calls [chime3D](#) to produce the html web page with a 3D graph.

### Value

html3D produces the pdb output file which can be read in Rasmol or other molecular structure viewers. html3D produces a html file with a 3D graph that can be rotated and manipulated in a web browser that supports the chime web browser plug-in.

### Note

Note chime is only available on windows or Mac OS currently. Using the chime plug-in on Linux is slightly complicated but is available if the CrossOver Plug-in is installed. Instructions on installing this and chime on Linux are available at [http://mirrors.rcsb.org/SMS/STINGm/help/chime\\_linux.html](http://mirrors.rcsb.org/SMS/STINGm/help/chime_linux.html)

If you wish to view a 3D graph in Rasmol, you will need to execute a Rasmol script similar to

```
load pdbfilename.pdb;
set axes on; select off;
connect;set ambient 40;
rotate x 180; select *;
spacefill 40
```

html3D calls [chime3D](#) to produce the html file from the pdb file.

The author would like to thank Willie Taylor, The National Institute for Medical Research, London, UK for help with the awk command on which this function is based.

### Author(s)

Aedin Culhane

## Examples

```

data(khan)
if (require(ade4, quiet = TRUE)) {
khan.bga<-bga(khan$train, khan$train.classes)
}

out.3D <-html3D(khan.bga$bet$ls, khan.bga$fac, writepdb=TRUE,
filenamebase ="Khan" , writehtml=TRUE)

## Not run:
browseURL(paste("file://", file.path(paste(getwd()),"/khan.html",
sep="")), sep="")

## End(Not run)

```

---

isDataFrame	<i>Converts microarray input data into a data frame suitable for analysis in ADE4.</i>
-------------	--

---

## Description

Converts input data into a data.frame suitable for analysis in ADE4. This function is called by bga and other made4 function

## Usage

```
isDataFrame(dataset, pos = FALSE, trans = FALSE)
```

## Arguments

dataset	A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
pos	Logical indicating whether to add an integer to dataset, to generate positive data.frame. Required for <code>dudi.coa</code> or <code>dudi.nsc</code> .
trans	Logical indicating whether dataset should be transposed. Default is FALSE.

## Details

[bga](#) and other functions in made4 call this function and it is generally **not** necessary to call `isDataFrame` this directly.

`isDataFrame` calls [asDataFrame](#), and will accept a [matrix](#), [data.frame](#), [ExpressionSet](#) or [marrayRaw-class](#) or `SummarizedExperiment` format. It will also transpose data or add a integer to generate a positive data matrix.

If the input data contains missing values (NA), these must first be removed or imputed (see the R libraries `impute()` or `pamr()`).



**Value**

Returns a `data.frame` suitable for analysis by `ade4` or `made4` functions.

**Author(s)**

Aedin Culhane

**See Also**

[as](#) in Bioconductor

**Examples**

```
data(geneData)
class(geneData)
dim(geneData)
dim(isDataFrame(geneData))
class(isDataFrame(geneData))
```

---

jmol3D	<i>Produce web graph of 3D graph that can be viewed using Jmol in web browser</i>
--------	---

---

**Description**

`jmol3D` produces a html web page with a 3D graph of a `pdb` file that can be rotated and manipulated in a web browser that supports `jmol`.

**Usage**

```
jmol3D(df, classvec=NULL, title=NULL, jmoldir="../jmol/Jmol.js",
       filename="output.html", point.size=40, xyz.axes =c(1:3), scaled=TRUE, ...)
```

**Arguments**

<code>df</code>	A <a href="#">matrix</a> or <a href="#">data.frame</a> containing the <code>x,y,z</code> coordinates. Typically the output from <a href="#">bga</a> such as the <code>\\$ls</code> or <code>\\$co</code> files, or other <code>xyz</code> coordinates ( <code>\\$li</code> or <code>\\$co</code> ) produced by <code>PCA</code> , <code>COA</code> or other <a href="#">dudi</a> .
<code>classvec</code>	Factor or vector which describes classes in the <code>df</code> . Default is <code>NULL</code> . If specified each group will be coloured in contrasting colours.
<code>title</code>	Character, the title (header) of the web page saved if <code>writehtml</code> is <code>TRUE</code> . The default is <code>NULL</code> .
<code>filename</code>	Character. The filename of the html to be saved. The default is <code>output.html</code> .
<code>xyz.axes</code>	Vector indicating which axes to use for <code>x</code> , <code>y</code> and <code>z</code> axes. By default, the first 3 columns of <code>df</code> .
<code>jmoldir</code>	Character indicating the local directory containing <code>jmol</code> .
<code>point.size</code>	Numeric, size of the points in plot. Default is 40.
<code>...</code>	Further arguments passed to or from other methods.

**Details**

Produces a html file, of a 3D graph which can be rotated using the opensource software jmol <http://jmol.sourceforge.net/> which is a java based software that is open source. For example see <http://bioinf.ucd.ie/people/aedin/jmoltest/test.html>.

**Value**

jmol3D output is a html file that can be viewed in any web browser that supports jmol <http://jmol.sourceforge.net/>.

**Note**

Note jmol must be downloaded on a local machine. Instructions on installing this are available at <http://jmol.sourceforge.net/>.

**Author(s)**

Aedin Culhane

---

khan	<i>Microarray gene expression dataset from Khan et al., 2001. Subset of 306 genes.</i>
------	--

---

**Description**

Khan contains gene expression profiles of four types of small round blue cell tumours of childhood (SRBCT) published by Khan et al. (2001). It also contains further gene annotation retrieved from SOURCE at <http://source.stanford.edu/>.

**Usage**

```
data(khan)
```

**Format**

Khan is dataset containing the following:

- `\$train:data.frame` of 306 rows and 64 columns. The training dataset of 64 arrays and 306 gene expression values
- `\$test:data.frame`, of 306 rows and 25 columns. The test dataset of 25 arrays and 306 genes expression values
- `\$gene.labels.imagesID`:vector of 306 Image clone identifiers corresponding to the rownames of `\$train` and `\$test`.
- `\$train.classes:factor` with 4 levels "EWS", "BL-NHL", "NB" and "RMS", which correspond to the four groups in the `\$train` dataset
- `\$test.classes:factor` with 5 levels "EWS", "BL-NHL", "NB", "RMS" and "Norm" which correspond to the five groups in the `\$test` dataset

- `data.frame` of 306 rows and 8 columns. This table contains further gene annotation retrieved from SOURCE <http://SOURCE.stanford.edu> in May 2004. For each of the 306 genes, it contains:
  - `CloneID` Clone ID
  - `UGCluster` The Unigene cluster to which the gene is assigned
  - `Symbol` The HUGO gene symbol
  - `LLID` The locus ID
  - `UGRepAcc` Nucleotide sequence accession number
  - `LLRepProtAcc` Protein sequence accession number
  - `Chromosome` chromosome location
  - `Cytoband` cytoband location

## Details

Khan et al., 2001 used cDNA microarrays containing 6567 clones of which 3789 were known genes and 2778 were ESTs to study the expression of genes in of four types of small round blue cell tumours of childhood (SRBCT). These were neuroblastoma (NB), rhabdomyosarcoma (RMS), Burkitt lymphoma, a subset of non-Hodgkin lymphoma (BL), and the Ewing family of tumours (EWS). Gene expression profiles from both tumour biopsy and cell line samples were obtained and are contained in this dataset. The dataset downloaded from the website contained the filtered dataset of 2308 gene expression profiles as described by Khan et al., 2001. This dataset is available from the <http://bioinf.ucd.ie/people/aedin/R/>.

In order to reduce the size of the MADE4 package, and produce small example datasets, the top 50 genes from the ends of 3 axes following bga were selected. This produced a reduced datasets of 306 genes.

## Source

khan contains a filtered data of 2308 gene expression profiles as published and provided by Khan et al. (2001) on the supplementary web site to their publication <http://research.nhgri.nih.gov/microarray/Supplement/>.

## References

Culhane AC, et al., 2002 Between-group analysis of microarray data. *Bioinformatics*. 18(12):1600-8.

Khan, J., Wei, J.S., Ringner, M., Saal, L.H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C.R., Peterson, C. et al. (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nat. Med.*, 7, 673-679.

## Examples

```
data(khan)
summary(khan)
```

## Description

NCI60 is a dataset of gene expression profiles of 60 National Cancer Institute (NCI) cell lines. These 60 human tumour cell lines are derived from patients with leukaemia, melanoma, along with lung, colon, central nervous system, ovarian, renal, breast and prostate cancers. This panel of cell lines have been subjected to several different DNA microarray studies using both Affymetrix and spotted cDNA array technology. This dataset contains subsets from one cDNA spotted (Ross et al., 2000) and one Affymetrix (Staunton et al., 2001) study, and are pre-processed as described by Culhane et al., 2003.

## Usage

```
data(NCI60)
```

## Format

The format is: List of 3

- `\$Ross:data.frame` containing 144 rows and 60 columns. 144 gene expression log ratio measurements of the NCI60 cell lines.
- `\$Affy:data.frame` containing 144 rows and 60 columns. 144 Affymetrix gene expression average difference measurements of the NCI60 cell lines.
- `\$classes:Data matrix` of 60 rows and 2 columns. The first column contains the names of the 60 cell line which were analysed. The second column lists the 9 phenotypes of the cell lines, which are BREAST, CNS, COLON, LEUK, MELAN, NSCLC, OVAR, PROSTATE, RENAL.
- `\$Annot:Data matrix` of 144 rows and 4 columns. The 144 rows contain the 144 genes in the `\$Ross` and `\$Affy` datasets, together with their Unigene IDs, and HUGO Gene Symbols. The Gene Symbols obtained for the `\$Ross` and `\$Affy` datasets differed (see note below), hence both are given. The columns of the `matrix` are the IMAGE ID of the clones of the `\$Ross` dataset, the HUGO Gene Symbols of these IMAGE clone ID obtained from SOURCE, the Affymetrix ID of the `\$Affy` dataset, and the HUGO Gene Symbols of these Affymetrix IDs obtained using `annaffy`.

## Details

The datasets were processed as described by Culhane et al., 2003.

The `Ross data.frame` contains gene expression profiles of each cell lines in the NCI-60 panel, which were determined using spotted cDNA arrays containing 9,703 human cDNAs (Ross et al., 2000). The data were downloaded from The NCI Genomics and Bioinformatics Group Datasets resource <http://discover.nci.nih.gov/datasetsNature2000.jsp>. The updated version of this dataset (updated 12/19/01) was retrieved. Data were provided as log ratio values.

In this study, rows (genes) with greater than 15 and were removed from analysis, reducing the dataset to 5643 spot values per cell line. Remaining missing values were imputed using a K nearest

neighbour method, with 16 neighbours and a Euclidean distance metric (Troyanskaya et al., 2001). The dataset `\$Ross` contains a subset of the 144 genes of the 1375 genes set described by Scherf et al., 2000. This datasets is available for download from <http://bioinf.ucd.ie/people/aedin/R/>.

In order to reduce the size of the example datasets, the Unigene ID's for each of the 1375 IMAGE ID's for these genes were obtained using SOURCE <http://source.stanford.edu>. These were compared with the Unigene ID's of the 1517 gene subset of the `\$Affy` dataset. 144 genes were common between the two datasets and these are contained in `\$Ross`.

The Affy data were derived using high density Hu6800 Affymetrix microarrays containing 7129 probe sets (Staunton et al., 2001). The dataset was downloaded from the Whitehead Institute Cancer Genomics supplemental data to the paper from Staunton et al., <http://www-genome.wi.mit.edu/mpr/NCI60/>, where the data were provided as average difference (perfect match-mismatch) values. As described by Staunton et al., an expression value of 100 units was assigned to all average difference values less than 100. Genes whose expression was invariant across all 60 cell lines were not considered, reducing the dataset to 4515 probe sets. This dataset `NCI60\$Affy` of 1517 probe set, contains genes in which the minimum change in gene expression across all 60 cell lines was greater than 500 average difference units. Data were logged (base 2) and median centred. This datasets is available for download from <http://bioinf.ucd.ie/people/aedin/R/>.

In order to reduce the size of the example datasets, the Unigene ID's for each of the 1517 Affymetrix ID of these genes were obtained using the function `aafUniGene` in the `annaffy` Bioconductor package. These 1517 Unigene IDs were compared with the Unigene ID's of the 1375 gene subset of the `\$Ross` dataset. 144 genes were common between the two datasets and these are contained in `\$Affy`.

## Source

These pre-processed datasets were available as a supplement to the paper:

Culhane AC, Perriere G, Higgins DG. Cross-platform comparison and visualisation of gene expression data using co-inertia analysis. *BMC Bioinformatics*. 2003 Nov 21;4(1):59. <http://www.biomedcentral.com/1471-2105/4/59>

## References

- Culhane AC, Perriere G, Higgins DG. Cross-platform comparison and visualisation of gene expression data using co-inertia analysis. *BMC Bioinformatics*. 2003 Nov 21;4(1):59.
- Ross DT, Scherf U, Eisen MB, Perou CM, Rees C, Spellman P, Iyer V, Jeffrey SS, Van de Rijn M, Waltham M, Pergamenschikov A, Lee JC, Lashkari D, Shalon D, Myers TG, Weinstein JN, Botstein D, Brown PO: Systematic variation in gene expression patterns in human cancer cell lines. *Nat Genet* 2000, 24:227-235
- Scherf U, Ross DT, Waltham M, Smith LH, Lee JK, Tanabe L, Kohn KW, Reinhold WC, Myers TG, Andrews DT, Scudiero DA, Eisen MB, Sausville EA, Pommier Y, Botstein D, Brown PO, Weinstein JN: A gene expression database for the molecular pharmacology of cancer. *Nat Genet* 2000, 24:236-244.
- Staunton JE, Slonim DK, Collier HA, Tamayo P, Angelo MJ, Park J, Scherf U, Lee JK, Reinhold WO, Weinstein JN, Mesirov JP, Lander ES, Golub TR: Chemosensitivity prediction by transcriptional profiling. *Proc Natl Acad Sci U S A* 2001, 98:10787-10792.

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB: Missing value estimation methods for DNA microarrays. *Bioinformatics* 2001, 17:520-525.

### Examples

```
data(NCI60)
summary(NCI60)
```

---

ord

*Ordination*

---

### Description

Run principal component analysis, correspondence analysis or non-symmetric correspondence analysis on gene expression data

### Usage

```
ord(dataset, type="coa", classvec=NULL, ord.nf=NULL, trans=FALSE, ...)
## S3 method for class 'ord'
plot(x, axis1=1, axis2=2, arraycol=NULL, genecol="gray25",
     nlab=10, genelabels= NULL, arraylabels=NULL, classvec=NULL, ...)
```

### Arguments

dataset	Training dataset. A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
classvec	A factor or vector which describes the classes in the training dataset.
type	Character, "coa", "pca" or "nsc" indicating which data transformation is required. The default value is type="coa".
ord.nf	Numeric. Indicating the number of eigenvector to be saved, by default, if NULL, all eigenvectors will be saved.
trans	Logical indicating whether 'dataset' should be transposed before ordination. Used by BGA Default is FALSE.
x	An object of class ord. The output from ord. It contains the projection coordinates from ord, the $\backslash$co$ or $\backslash$li$ coordinates to be plotted.
arraycol, genecol	Character, colour of points on plot. If arraycol is NULL, arraycol will obtain a set of contrasting colours using <a href="#">getcol</a> , for each classes of cases (microarray samples) on the array (case) plot. genecol is the colour of the points for each variable (genes) on gene plot.
nlab	Numeric. An integer indicating the number of variables (genes) at the end of axes to be labelled, on the gene plot.

<code>axis1</code>	Integer, the column number for the x-axis. The default is 1.
<code>axis2</code>	Integer, the column number for the y-axis, The default is 2.
<code>genelabels</code>	A vector of variables labels, if <code>genelabels=NULL</code> the <code>row.names</code> of input matrix dataset will be used.
<code>arraylabels</code>	A vector of variables labels, if <code>arraylabels=NULL</code> the <code>col.names</code> of input matrix dataset will be used.
<code>...</code>	further arguments passed to or from other methods.

## Details

`ord` calls either `dudi.pca`, `dudi.coa` or `dudi.nsc` on the input dataset. The input format of the dataset is verified using `isDataFrame`.

If the user defines microarray sample groupings, these are colours on plots produced by `plot.ord`.

### Plotting and visualising bga results:

*2D plots:* `plotarrays` to draw an xy plot of cases ( $\backslash\$i$ s). `plotgenes`, is used to draw an xy plot of the variables (genes).

*3D plots:* 3D graphs can be generated using `do3D` and `html3D`. `html3D` produces a web page in which a 3D plot can be interactively rotated, zoomed, and in which classes or groups of cases can be easily highlighted.

*1D plots, show one axis only:* 1D graphs can be plotted using `graph1D`. `graph1D` can be used to plot either cases (microarrays) or variables (genes) and only requires a vector of coordinates ( $\backslash\$i$ ,  $\backslash\$co$ )

### Analysis of the distribution of variance among axes:

The number of axes or principal components from a `ord` will equal `nrow` the number of rows, or the `ncol`, number of columns of the dataset (whichever is less).

The distribution of variance among axes is described in the eigenvalues ( $\backslash\$eig$ ) of the `ord` analysis. These can be visualised using a scree plot, using `scatterutil.eigen` as it done in `plot.ord`. It is also useful to visualise the principal components from a using a `ord` or principal components analysis `dudi.pca`, or correspondence analysis `dudi.coa` using a heatmap. In MADE4 the function `heatmap` will plot a heatmap with nicer default colours.

### Extracting list of top variables (genes):

Use `topgenes` to get list of variables or cases at the ends of axes. It will return a list of the top `n` variables (by default `n=5`) at the positive, negative or both ends of an axes. `sumstats` can be used to return the angle (slope) and distance from the origin of a list of coordinates.

## Value

A list with a class `ord` containing:

<code>ord</code>	Results of initial ordination. A list of class "dudi" (see <code>dudi</code> )
<code>fac</code>	The input <code>classvec</code> , the factor or vector which described the classes in the input dataset. Can be <code>NULL</code> .

**Author(s)**

Aedin Culhane

**See Also**See Also [dudi.pca](#), [dudi.coa](#) or [dudi.nsc](#), [bga](#),**Examples**

```

data(khan)

if (require(ade4, quiet = TRUE)) {
  khan.coa<-ord(khan$train, classvec=khan$train.classes, type="coa")
}

khan.coa
plot(khan.coa, genelabels=khan$annotation$Symbol)
plotarrays(khan.coa)
# Provide a view of the first 5 principal components (axes) of the correspondence analysis
heatmap(khan.coa$ord$co[,1:5], dend="none", dualScale=FALSE)

```

---

 overview

---

*Draw boxplot, histogram and hierarchical tree of gene expression data*


---

**Description**

Very simple wrapper function that draws a boxplot, histogram and hierarchical tree of expression data

**Usage**

```

overview(dataset, labels = NULL, title = "", classvec = NULL,
  hc = TRUE, boxplot = TRUE, hist = TRUE, returnTree=FALSE)

```

**Arguments**

dataset	A <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
labels	Vector, labels to be placed on samples in plots. Default is <code>rownames(dataset)</code> .
title	Character, label to be placed on plots. Default is <code>NULL</code> .
classvec	A factor or vector which describes the classes in columns of the dataset. Default is <code>NULL</code> . If included columns (array samples) on the dendrogram will be coloured by class.
hc	Logical. Draw dendrogram of hierarchical cluster analysis of cases. Default is <code>TRUE</code> .



boxplot	Logical. Draw boxplot. Default is TRUE.
hist	Logical. Draw histogram. Default is TRUE.
returnTree	Logical. Return the hierarchical cluster analysis results. Default is FALSE.

### Details

The hierarchical plot is produced using average linkage cluster analysis with Pearson's correlation metric as described by Eisen et al.,1999.

### Author(s)

Aedin Culhane

### See Also

See also as [boxplot](#), [hclust](#), [hist](#)

### Examples

```
data(khan)

logkhan<-log2(khan$train)
print(class(logkhan))
overview(logkhan, title="Subset of Khan Train")

overview(logkhan, classvec=khan$train.classes,
labels=khan$train.classes,title="Subset of Khan Train")

overview(logkhan, classvec=khan$train.classes,
labels=khan$train.classes,title="Subset of Khan Train", boxplot=FALSE,
his=FALSE)
```

---

plotarrays	<i>Graph xy plot of variable (array) projections from ordination, between group analysis or coinertia analysis.</i>
------------	---

---

### Description

Graph xy plot of variables using s.var, s.groups or s.match.col. Useful for visualising array coordinates ( $\lambda_i$ ) resulting from ord, bga or cia of microarray data.

### Usage

```
plotarrays(coord, axis1 = 1, axis2 = 2, arraylabels = NULL, classvec=NULL,
graph = c("groups", "simple", "labels", "groups2", "coinertia","coinertia2"),
labelsize=1, star=1, ellipse=1, arraycol=NULL, ...)
```

**Arguments**

coord	a <a href="#">data.frame</a> or <a href="#">matrix</a> or object from ord bga or cia analysis with at least two columns, containing x, y coordinates to be plotted
axis1	An integer, the column number for the x-axis. Default is 1, so axes 1 is <code>dudi-var[,1]</code>
axis2	An integer, the column number for the y-axis. Default is 2, so axes 2 is <code>dudi-var[,2]</code>
arraylabels	A vector of variables labels. Default is <code>row.names(coord)</code>
classvec	A factor or vector which describes the classes in coord. Default is NULL. If included variables will be coloured by class.
graph	A character of type "groups", "simple", "labels", "groups2", "coinertia" or "coinertia2" which specifies the type of plot type or "graph" to be drawn. By default the graph will be selected depending on the class of coord, and whether a classvector is specified
labelsize	Size of sample labels, by default=1
star	If drawing groups, whether to join samples to centroid creating a "star"
ellipse	If drawing groups, whether to draw an ellipse or ring around the samples
arraycol	Character with length equal to the number of levels in the factor classvec. Colors for each of the levels in the factor classvec
...	further arguments passed to or from other method

**Details**

plotarrays calls the function `s.var`, `s.groups` or `s.match.col`.

If you wish to return a table or list of the top array at the end of an axis, use the function [topgenes](#).

**Value**

An xy plot

**Note**

plotarrays plots variables using `s.var`, `s.groups`, `s.match.col` which are modifieds version of `s.label`, `s.class`, and `s.match`.

**Author(s)**

Aedin Culhane

**See Also**

See Also as [s.var](#) and [s.label](#)

**Examples**

```

data(khan)
if (require(ade4, quiet = TRUE)) {
khan.bga<-bga(khan$train, khan$train.classes)
}
attach(khan.bga)
par(mfrow=c(2,1))
plotarrays(khan.bga)
plotarrays(khan.bga, graph="simple")
plotarrays(khan.bga, graph="labels")
plotarrays(khan.bga, graph="groups")
plotarrays(khan.bga, graph="groups2")

```

---

plotgenes	<i>Graph xy plot of variable (gene) projections from PCA or COA. Only label variables at ends of axes</i>
-----------	---

---

**Description**

Graph xy plot of variables but only label variables at ends of X and Y axes. Useful for graphing genes coordinates ( $\$co$ ) resulting from PCA or COA of microarray data.

**Usage**

```

plotgenes(coord, nlab = 10, axis1 = 1, axis2 = 2, genelabels =
row.names(coord), boxes = TRUE, colpoints = "black", ...)

```

**Arguments**

coord	a <a href="#">data.frame</a> or <a href="#">matrix</a> or object from ord bga or cia analysis with at least two columns, containing x, y coordinates to be plotted.
nlab	Numeric. An integer indicating the number of variables at ends of axes to be labelled.
axis1	An integer, the column number for the x-axis. Default is 1, so axis 1 is dudi-var[,1].
axis2	An integer, the column number for the y-axis. Default is 2, so axis 2 is dudi-var[,2].
genelabels	A vector of gene (variable) labels. Default is row.names(coord)
boxes	A logical, indicating whether a box should be plotted surrounding each variable label. The default is TRUE.
colpoints	The colour of the points on the plot. The default is "black".
...	further arguments passed to or from other method.

**Details**

plotgenes calls the function genes which return an index of the "top" variables at the ends of the x and y axes.

If you wish to return a table or list of the top genes at the end of an axis, use the function [topgenes](#).

**Value**

An xy plot

**Note**

plotgenes plots variables using [s.var](#), which is a modified version of [s.label](#).

**Author(s)**

Aedin Culhane

**See Also**

See Also as [s.var](#) and [s.label](#)

**Examples**

```
data(khan)
if (require(ade4, quiet = TRUE)) {
khan.ord<-ord(khan$train, classvec=khan$train.classes)
}
par(mfrow=c(2,2))
#s.var(khan.ord$co, col=as.numeric(khan$train.classes), clabel=0.8)
plotgenes(khan.ord, colpoints="red")
plotgenes(khan.ord, colpoints="red", genelabels=khan$annotation$Symbol)
plotgenes(khan.ord, colpoints="gray", genelabels=khan$annotation$Symbol,boxes=FALSE)
```

---

```
prettyDend
```

*Draw hierarchical tree of gene expression data with a colorbar for numerous class vectors*

---

**Description**

Function which performs a hierarchical cluster analysis of data, drawing a dendrogram, with colorbars for different sample covariate beneath the dendrogram

**Usage**

```
prettyDend(dataset, labels = NULL, title = "", classvec = NULL,
covars=1, returnTree=FALSE, getPalette=getcol,...)
```

**Arguments**

dataset	a <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> . If the input is gene expression data in a <a href="#">matrix</a> or <a href="#">data.frame</a> . The rows and columns are expected to contain the variables (genes) and cases (array samples) respectively.
labels	Vector, labels to be placed on samples in plots. Default is rownames(dataset).
title	Character, label to be placed on plots. Default is NULL.
classvec	A factor or vector or matrix or data.frame which describes the classes in columns of the dataset. Default is NULL.
covars	Numeric. The columns of the data.frame classve to be used as class vectors. These will be displayed as color bars under the dendrogram. The default is 1 (column 1).
returnTree	Logical. Return the hierarchical cluster analysis results. Default is FALSE.
getPalette	Function, which generates a palette of colors. The default uses getcol function in made4. Other examples are provided below
...	further arguments passed to or from other methods.

**Details**

The hierarchical plot is produced using average linkage cluster analysis with 1- Pearson's correlation metric. The default set of colors used to generate the color bars of the plots can be changed (see example below). By default, if there is only two levels in the factor, the colors will be black and grey.

**Author(s)**

Aedin Culhane

**See Also**

See also as [overview](#), [hclust](#)

**Examples**

```
data(khan)
logkhan<-log2(khan$train)

# Get a character vector which defines which khan samples are cell lines or tissue sample
khanAnnot= cbind(as.character(khan$train.classes),khan$cellType)
print(khanAnnot[1:3,])

# Add 2 color bar, one for cancer subtype, another for cell type under dendrogram
prettyDend(logkhan, classvec=khanAnnot, covars = c(1,2), labels=khan$train.classes)

# To change the palette of colors
# Use topo.colors(), see colors() for more help on inbuilt palettes

prettyDend(logkhan, classvec=khanAnnot, covars = c(1,2),
```

```

labels=khan$train.classes, getPalette=topo.colors)

# To use RColorBrewer Palettes
library(RColorBrewer)

# Use RColorBrewer Dark2 which contains 8 colors
prettyDend(logkhan, classvec=khanAnnot, covars = c(1,2),
labels=khan$train.classes, getPalette=function(x) brewer.pal(8,"Dark2")[1:x])

# Use RColorBrewer Set1 which contains 9 colors
prettyDend(logkhan, classvec=khanAnnot, covars = c(1,2),
labels=khan$train.classes, getPalette=function(x) brewer.pal(9,"Set1")[1:x])

```

---

randomiser

*Randomly reassign training and test samples*


---

### Description

This function is used to check for bias between a training and test data. It return a new index, which randomly re-assigns samples in the training data to the test dataset and vice versa.

### Usage

```
randomiser(ntrain = 77, ntest = 19)
```

### Arguments

ntrain	Numeric. A integer indicating the number of cases in the training dataset
ntest	Numeric. A integer indicating the number of cases in the test dataset

### Details

Produces new indices that can be used for training/test datasets

### Value

It returns a list, containing 2 vectors

train	A vector of length ntrain, which can be used to index a new training dataset
test	A vector of length ntest, which can be used to index a new test dataset

### Author(s)

Aedin Culhane

**Examples**

```

randomiser(10,5)
train<-matrix(rnorm(400), ncol=20, nrow=20, dimnames=list(1:20,
paste("train",letters[1:20], sep=".")))
test<-matrix(rnorm(200), ncol=10, nrow=20, dimnames=list(1:20,
paste("test",LETTERS[1:10], sep=".")))
all<-cbind(train,test)

colnames(train)
colnames(test)
newInd<-randomiser(ntrain=20, ntest=10)

newtrain<-all[,newInd$train]
newtest<-all[,newInd$test]

colnames(newtrain)
colnames(newtest)

```

---

s.var

*Draw xy plots of results of multivariate analysis*


---

**Description**

A number of functions for graphically displaying results from between group, coinertia, or other multivariate analysis of gene expression data. s.var, s.groups and s.match.col are derived from ADE4 graphics modules, s.label, s.class and s.match respectively.

**Usage**

```

s.var(dfxy, xax = 1, yax = 2, label = row.names(dfxy), clabel = 1, pch = 20,
      cpoint = if (clabel == 0) 1 else 0, neig = NULL, cneig = 2, xlim = NULL,
      ylim = NULL, grid = TRUE, addaxes = TRUE, cgrid = 1, include.origin = TRUE,
      origin = c(0, 0), sub = "", csub = 1.25, possub = "bottomleft", pixmap = NULL,
      contour = NULL, area = NULL, add.plot = FALSE, colpoints = rep(1, nrow(dfxy)),
      boxes = TRUE, ...)
s.groups(dfxy, classvec, wt = rep(1, length(classvec)), xax = 1, yax = 2, cstar = 1,
          cellipse = 1.5, axesell = TRUE, label = levels(classvec), slabel = row.names(dfxy),
          clabel = 1, cpoint = 1, pch = 20, col = rep(1, length(levels(classvec))),
          xlim = NULL, ylim = NULL, grid = TRUE, addaxes = TRUE, origin = c(0, 0),
          include.origin = TRUE, sub = "", csub = 1, possub = "bottomleft",
          cgrid = 1, pixmap = NULL, contour = NULL, area = NULL, add.plot = FALSE, ...)
s.match.col(df1xy, df2xy, xax = 1, yax = 2, pch = 20, cpoint = 1,
            label = row.names(df1xy), clabel = 1, edge = TRUE, xlim = NULL, ylim = NULL,
            grid = TRUE, addaxes = TRUE, cgrid = 1, include.origin = TRUE, origin = c(0, 0),
            sub = "", csub = 1.25, possub = "bottomleft", pixmap = NULL, contour = NULL,
            area = NULL, add.plot = FALSE, col = rep(1, nrow(df1xy)), classvec=NULL,...)

```

**Arguments**

dfxy	coordinates, normally the $\$ls$ , $\$co$ or $\$li$ coordinates from PCA, COA or BGA with at least two columns, containing the x,y coordinates to be plotted
df1xy, df2xy	coordinates from cia, normally the $\$mX$ , $\$mY$ coordinates with at least two columns, containing the x,y coordinates to be plotted
colpoints	Character, colour of arrays (cases),genes or points
xax	the column number for the x-axis
yax	the column number for the y-axis
classvec	Factor. A factor, classvec partitioning the rows of the data frame in classes
boxes	A logical, indicating whether a box should be drawn around each variable label. The default is TRUE
add.plot	logical if TRUE plot draw in current graphics window
origin, sub, wt, cstar, cellipse, axesell, col, edge	Graphing parameters. See <code>s.label</code> or <code>s.class</code> for details
clabel, pch, cpoint, neig, cneig, xlim, ylim, grid, addaxes, cgrid, include.origin, csub, possub, pixmap, contour, area	Graphing parameters, see <code>s.match</code> for details
...	further arguments passed to or from other method

**Details**

These functions are modified versions of `s.label`, `s.class` and `s.match` graph modules from ADE4. Please see [s.label](#) or [s.class](#) and [s.match](#) for father details.

These functions `s.var`, `s.groups` and `s.match.col` are called by [plot.bga](#) and [plot.cia](#).

**Author(s)**

Aedin Culhane

**See Also**

See also [s.class](#), [s.label](#), [s.match](#)

**Examples**

```
data(khan)
if (require(ade4, quiet = TRUE)) {
khan.bga<-bga(khan$train, khan$train.classes)
attach(khan.bga)
par(mfrow=c(2,2))
s.var(bet$ls, col=as.numeric(khan$train.classes), clabel=0.8)
s.groups(bet$ls, khan$train.classes, add.plot=TRUE,
col=c(1:length(levels(khan$train.classes))) )
}
```



---

sumstats	<i>Summary statistics on xy co-ordinates, returns the slopes and distance from origin of each co-ordinate.</i>
----------	--

---

### Description

Given a `data.frame` or `matrix` containing xy coordinates, it returns the slope and distance from origin of each coordinate.

### Usage

```
sumstats(array, xax = 1, yax = 2)
```

### Arguments

array	A <code>data.frame</code> or <code>matrix</code> containing xy coordinates, normally a <code>\\$co</code> , <code>\\$li</code> from <code>dudi</code> such as PCA or COA, or <code>\\$ls</code> from <code>bga</code>
xax	Numeric, an integer indicating the column of the x axis coordinates. Default <code>xax=1</code>
yax	Numeric, an integer indicating the column of the y axis coordinates. Default <code>yax=2</code>

### Details

In PCA or COA, the variables (upregulated genes) that are most associated with a case (microarray sample), are those that are projected in the same direction from the origin.

Variables or cases that have a greater contribution to the variance in the data are projected further from the origin in PCA. Equally variables and cases with the strong association have a high chi-square value, and are projected with greater distance from the origin in COA, See a description from Culhane et al., 2002 for more details.

Although the projection of co-ordinates are best visualised on an xy plot, `sumstats` returns the slope and distance from origin of each x,y coordinate in a matrix.

### Value

A matrix (`ncol=3`) containing

```
slope
angle (in degrees)
distance from origin
```

of each x,y coordinates in a matrix.

### Author(s)

Aedin Culhane

**Examples**

```

data(khan)

if (require(ade4, quiet = TRUE)) {

khan.bga<-bga(khan$train, khan$train.classes)}

plotarrays(khan.bga$bet$ls, classvec=khan$train.classes)
st.out<-sumstats(khan.bga$bet$ls)

# Get stats on classes EWS and BL
EWS<-khan$train.classes==levels(khan$train.classes)[1]
st.out[EWS,]

BL<-khan$train.classes==levels(khan$train.classes)[2]
st.out[BL,]

# Add dashed line to plot to highlight min and max slopes of class BL
slope.BL.min<-min(st.out[BL,1])
slope.BL.max<-max(st.out[BL,1])
abline(c(0,slope.BL.min), col="red", lty=5)
abline(c(0,slope.BL.max), col="red", lty=5)

```

---

suppl

*Projection of supplementary data onto axes from a between group analysis*


---

**Description**

Projection and class prediction of supplementary points onto axes from a between group analysis, [bga](#).

**Usage**

```

suppl(dudi.bga, supdata, supvec = NULL, assign=TRUE, ...)
## S3 method for class 'suppl'
plot(x, dudi.bga, axis1=1, axis2=2, supvec=x$true.class,
supvec.pred= x$predicted, ...)

```

**Arguments**

dudi.bga	An object returned by <a href="#">bga</a> .
supdata	Test or blind dataset. Accepted formats are a <a href="#">matrix</a> , <a href="#">data.frame</a> , <a href="#">ExpressionSet</a> or <a href="#">marrayRaw-class</a> .
supvec	A factor or vector which describes the classes in the training dataset.
supvec.pred	A factor or vector which describes the classes which were predicted by <a href="#">suppl</a> .

<code>assign</code>	A logical indicating whether class assignment should be calculated using the method described by Culhane et al., 2002. The default value is TRUE.
<code>x</code>	An object returned by <a href="#">suppl</a> .
<code>axis1</code>	Integer, the column number for the x-axis. The default is 1.
<code>axis2</code>	Integer, the column number for the y-axis. The default is 2.
<code>...</code>	further arguments passed to or from other methods.

### Details

After performing a between group analysis on a training dataset using [bga](#), a test dataset can be then projected onto bga axes using [suppl](#).

`suppl` returns the projected coordinates and assignment of each test case (array).

The test dataset must contain the same number of variables (genes) as the training dataset. The input format of both the training dataset and test dataset are verified using `isDataFrame`. Use [plot.bga](#) to plot results from `bga`.

### Value

A list containing:

`suppl` An object returned by [suppl](#)

### Author(s)

Aedin Culhane

### References

Culhane AC, et al., 2002 Between-group analysis of microarray data. *Bioinformatics*. 18(12):1600-8.

### See Also

See Also [bga](#), [bca](#), [plot.bga](#), [bga.jackknife](#)

### Examples

```
data(khan)
#khan.bga<-bga(khan$train, khan$train.classes)
if (require(ade4, quiet = TRUE)) {
khan.bga<-bga.suppl(khan$train, supdata=khan$test, classvec=khan$train.classes,
supvec=khan$test.classes)

khan.bga
plot.bga(khan.bga, genelabels=khan$annotation$Symbol)
khan.bga$suppl

plot.suppl(khan.bga$suppl, khan.bga)
plot.suppl(khan.bga$suppl, khan.bga, supvec=NULL, supvec.pred=NULL)
```

```
plot.suppl(khan.bga$suppl, khan.bga, axis1=2, axis2=3, supvec=NULL, supvec.pred=NULL)
}
```

---

topgenes	<i>Topgenes, returns a list of variables at the ends (positive, negative or both) of an axis</i>
----------	--

---

### Description

topgenes will return a list of the top N variables from the positive, negative or both ends of an axis. That is, it returns a list of variables that have the maximum and/or minimum values in a vector.

### Usage

```
topgenes(x, n = 10, axis = 1, labels = row.names(x), ends = "both", ...)
```

### Arguments

x	A <a href="#">vector</a> , <a href="#">matrix</a> or <a href="#">data.frame</a> . Typically a data frame <code>\\$co</code> or <code>\\$li</code> from <a href="#">dudi</a> or <code>\\$ls</code> , <code>\\$li</code> , <code>\\$co</code> from <a href="#">bga</a> .
n	An integer indicating the number of variables to be returned. Default is 5.
axis	An integer indicating the column of x. Default is 1 (first axis, of <code>\\$co</code> or <code>\\$li</code> file)
labels	A vector of row names, for <code>x[,axis]</code> . Default values is <code>row.names(x)</code>
ends	A string, "both", "neg", "pos", indicating whether variable label should be return from both, the negative or the positive end of an axis. The default is both.
...	further arguments passed to or from other methods

### Details

topgenes calls [genes1d](#). `genes1d` is similar to [genes](#), but returns an index of genes at the ends of one axis not two axes. Given a `\$co` or `\$li` file it will return that variables at the ends of the axis.

### Value

Returns a vector or list of vectors.

### Author(s)

AedinCulhane

### See Also

See Also as [genes](#)

**Examples**

```
# Simple example
a<-rnorm(50)
order(a)
topgenes(a, labels=c(1:length(a)), ends="neg")

# Applied example
data(khan)
if (require(ade4, quiet = TRUE)) {
khan.coa<-ord(khan$train[1:100,])
ind<-topgenes(khan.coa, ends="pos")
ind.ID<-topgenes(khan.coa, ends="pos", labels=khan$gene.labels.imagesID)
ind.symbol<-topgenes(khan.coa, ends="pos", labels=khan$annotation$Symbol)
Top10.pos<- cbind("Gene Symbol"=ind.symbol,
                 "Clone ID"=ind.ID, "Coordinates"=khan.coa$ord$li[ind,], row.names=c(1:length(ind)))
Top10.pos
```

# Index

- \* **color**
  - getcol, 25
- \* **datasets**
  - khan, 34
  - NCI60, 36
- \* **hplot**
  - between.graph, 5
  - cia, 14
  - commonMap, 16
  - do3d, 19
  - getcol, 25
  - graph1D, 26
  - heatplot, 27
  - html3D, 30
  - overview, 40
  - plotarrays, 41
  - plotgenes, 43
  - prettyDend, 44
- \* **internal**
  - asDataFrame, 2
  - checkfac, 12
  - chime3D, 12
  - dudi.rwcoa, 21
  - forrwcoa, 22
  - genes, 23
  - genes1d, 23
  - jmol3D, 33
  - s.var, 47
- \* **manip**
  - bet.coinertia, 3
  - bga, 6
  - bga.jackknife, 9
  - bga.suppl, 10
  - comparelists, 18
  - graph1D, 26
  - heatplot, 27
  - isDataFrame, 32
  - ord, 38
  - overview, 40
  - prettyDend, 44
  - randomiser, 46
  - sumstats, 49
  - suppl, 50
  - topgenes, 52
- \* **multivariate**
  - bet.coinertia, 3
  - between.graph, 5
  - bga, 6
  - bga.jackknife, 9
  - bga.suppl, 10
  - cia, 14
  - commonMap, 16
  - ord, 38
  - plotarrays, 41
  - plotgenes, 43
  - s.var, 47
  - suppl, 50
- as, 33
- asDataFrame, 2, 32
- bca, 4, 7–9, 11, 51
- bet.coinertia, 3, 21, 22
- between.graph, 5, 7, 17
- bga, 3–5, 6, 8, 9, 11, 15, 19, 24, 30, 32, 33, 40, 49–52
- bga.jackknife, 7, 8, 9, 11, 51
- bga.suppl, 6, 7, 9, 10
- boxplot, 41
- checkfac, 12
- chime3D, 12, 31
- cia, 3, 4, 14, 24
- COA, 21, 22
- coinertia, 4, 5, 15, 16
- commonMap, 16
- comparelists, 18
- data.frame, 3, 4, 6, 9, 10, 14, 19, 24, 26, 27,

- [30, 32–36, 38, 40, 42, 43, 45, 49, 50, 52](#)
- [dendrogram, 29](#)
- [do3D, 7, 39](#)
- [do3d, 19](#)
- [dudi, 4, 8, 11, 16, 21, 22, 24, 30, 33, 39, 49, 52](#)
- [dudi.coa, 5, 8, 14, 16, 19, 21, 22, 26, 39, 40](#)
- [dudi.nsc, 4, 14, 16, 39, 40](#)
- [dudi.pca, 4, 5, 8, 19, 21, 22, 26, 39, 40](#)
- [dudi.rwcoa, 14, 21](#)
- [ExpressionSet, 3, 4, 6, 10, 14, 27, 32, 38, 40, 45, 50](#)
- [factor, 12, 34](#)
- [forrwcoa, 22](#)
- [genes, 23, 24, 52](#)
- [genes1d, 23, 52](#)
- [getcol, 5, 25](#)
- [graph1D, 5–7, 17, 26, 39](#)
- [hclust, 29, 41, 45](#)
- [heatmap, 29](#)
- [heatmap.2, 28](#)
- [heatplot, 8, 27, 39](#)
- [hist, 41](#)
- [html3D, 7, 30, 39](#)
- [intersect, 18](#)
- [isDataFrame, 3, 7, 32, 39](#)
- [jmol3D, 33](#)
- [khan, 34](#)
- [matrix, 3, 4, 6, 9, 10, 14, 19, 24, 26, 27, 30, 32, 33, 36, 38, 40, 42, 43, 45, 49, 50, 52](#)
- [NCI60, 36](#)
- [ord, 38](#)
- [overview, 40, 45](#)
- [plot.bga, 9, 11, 48, 51](#)
- [plot.bga \(bga\), 6](#)
- [plot.cia, 48](#)
- [plot.cia \(cia\), 14](#)
- [plot.ord \(ord\), 38](#)
- [plot.suppl \(suppl\), 50](#)
- [plotarrays, 7, 39, 41](#)
- [plotgenes, 7, 23, 39, 43](#)
- [prettyDend, 44](#)
- [print.comparelists \(comparelists\), 18](#)
- [randomiser, 7, 46](#)
- [rotate3d \(do3d\), 19](#)
- [s.class, 42, 48](#)
- [s.groups, 42](#)
- [s.groups \(s.var\), 47](#)
- [s.label, 42, 44, 48](#)
- [s.match, 42, 48](#)
- [s.match.col, 42](#)
- [s.match.col \(s.var\), 47](#)
- [s.var, 42, 44, 47](#)
- [scatterplot3d, 19, 20](#)
- [scatterutil.eigen, 8, 39](#)
- [setdiff, 18](#)
- [sumstats, 8, 39, 49](#)
- [suppl, 7–9, 11, 50, 50, 51](#)
- [suppl.bga, 8](#)
- [topgenes, 8, 23, 39, 42, 44, 52](#)
- [vector, 12, 26, 52](#)