Package 'bios2mds'

August 21, 2026

Title From Biological Sequences to Multidimensional Scaling

Version 1.2.4

Date 2026-08-11

Author Julien Pele [aut],
Jean-Michel Becu [aut],
Rym Ben Boubaker [ctb],
Herve Abdi [ctb],
Marie Chabbert [cre]

Maintainer Marie Chabbert <marie.chabbert@univ-angers.fr>

Depends R (>= 3.6.0), amap, e1071, scales, cluster, rgl

Description Utilities dedicated to the analysis of
biological sequences by metric MultiDimensional Scaling with
projection of supplementary data. It contains functions for
reading multiple sequence alignment files, calculating distance
matrices, performing metric multidimensional scaling and
visualizing results.

License GPL

LazyData true

Repository CRAN

Date/Publication 2026-08-21 17:40:08 UTC

NeedsCompilation no

Contents

bios2mds-package 2
col.group 5
dif 6
dis 7
export.fasta 9
extract.cluster 10
gpcr 11
import.fasta 13

import.msf . . . 15
kmeans.run . . . 16
mat.dif . . . 17
mat.dis . . . 18
mmds . . . 20
mmds.2D.multi . . . 22
mmds.2D.plot . . . 25
mmds.3D.plot . . . 29
mmds.plot . . . 31
mmds.project . . . 32
random.msa . . . 35
scree.plot . . . 36
sil.score . . . 37
sub.mat . . . 39
write.mmds.pdb . . . 41

Index **43**

`bios2mds-package` *From BIOlogical Sequences to MultiDimensional Scaling*

Description

The `bios2mds` package has been developed in the Bioinformatics team at MITOVASC Laboratory, UMR CNRS 6015 / INSERM 1083, University of Angers - FRANCE.

This package is dedicated to the analysis of biological sequences by metric MultiDimensional Scaling (MDS) with projection of supplementary data. It contains functions for reading multiple sequence alignment (MSA) files, calculating distance matrices from MSA files, performing MDS analysis and visualizing results. The MDS analysis and visualization tools can be applied to any kind of data.

The main functionalities of `bios2mds` are summarized below:

(1) BUILDING DISTANCE MATRICES FROM MULTIPLE SEQUENCE ALIGNMENTS : Several functions allow users to read multiple sequence alignment files and to compute matrices of distances between these sequences:

- `import.fasta`: reads a multiple sequence alignment in FASTA format.
- `import.msf`: reads a multiple sequence alignment in MSF format.
- `mat.dif`: computes a matrix of pairwise distances between sequences based on sequence difference.
- `mat.dis`: computes a matrix of pairwise distances between sequences based on sequence dissimilarity.

(2) MULTIDIMENSIONAL SCALING : A function performs metric MDS analysis of a distance matrix between active elements with the option of projecting supplementary elements onto the active space.

- `mmds`: performs metric multidimensional scaling.
- `mmds.project`: performs projection of supplementary elements onto the active space.

(3) GRAPHICAL TOOLS : Several functions are proposed to visualize results of metric MDS analysis:

- `scree.plot`: draws the scree plot of eigenvalues.
- `mmds.2D.plot`: draws a scatter plot of the MDS coordinates.
- `mmds.2D.multi`: draws a scatter plot of the MDS coordinates with projection of multiple groups of supplementary element.
- `mmds.3D.plot`: Displays a 3D plot of the MDS coordinates.
- `mmds.plot`: wrapper function that draws the scree plot and three scatter plots of MDS coordinates.
- `col.group`: colors scatter plots with user provided groupings and colors.
- `write.mmds.pdb`: writes MDS coordinates in a PDB formatted file for 3D visualisation.

(4) CLUSTER ANALYSIS : Several functions allow users to perform data clustering and to assess the clustering robustness:

- `kmeans.run`: performs multiple runs of K-means clustering and analyzes clusters
- `sil.score`: calculates the silhouette score from multiple K-means runs to determine the optimal number of clusters.
- `extract.cluster`: extraction of clusters alignments

(5) DATASETS : Two raw datasets are proposed to test the functionalities of `bios2mds`. They correspond to the multiple sequence alignments of GPCRs from *H. sapiens* and *D. melanogaster* in .msf and .fa formats (msa/human_gpcr.* and msa/drome_gpcr.*). They are based on the non-redundant sets of non-olfactory class A GPCRs, prepared and analyzed in Deville et al. (2009) and updated with the July 2009 release of Uniprot `https://www.uniprot.org`. Each MSA file is related to a .csv file that assigns a group and a color to each sequence of the alignment (csv/human_gpcr_group.csv and csv/drome_gpcr_group.csv).

Pre-analyzed data from these two MSA files are in `gpcr`. Moreover, `gpcr` contain projection of GPCRs from *N. vectensis* and from *C. intestinalis* onto the active space of human GPCRs calculated by MDS analysis.

For an index of functions, use library(help = `bios2mds`).

Details

Package	bios2mds
Type	Package
Version	1.2.4
Repository	CRAN
Date	2026-08-11
License	GPL version 2 or newer

Author(s)

Julien Pele [aut], Jean-Michel Becu [aut], Rym Ben Boubaker [ctb], Herve Abdi [ctb], Marie Chabbert [cre]. Maintainer: Marie Chabbert <marie.chabbert@univ-angers.fr>

References

citation('bios2mds')

Examples

```
# The MSA files provided with the package correspond to the sequence
# alignment of non-olfactory class A G-protein-coupled receptors from
# H. sapiens and D. melanogaster prepared by Deville et al. (2009).
# loading GPCR data
wd <- tempdir()
file <- file.path(wd,"R.pdb")
data(gpcr)

# building distance matrices between the aligned GPCR sequences from
# H. sapiens and D. melanogaster
human <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
drome <- import.fasta(system.file("msa/drome_gpcr.fa", package = "bios2mds"))

#active <- mat.dif(human, human)
# or
active <- gpcr$dif$sapiens.sapiens

#sup <- mat.dif(drome, human)
# or
sup <- gpcr$dif$melanogaster.sapiens

# performing MDS analysis of the GPCR sequences from H. sapiens
mmds_active <- mmds(active, group.file=system.file(
"csv/human_gpcr_group.csv",package = "bios2mds"))

# performing MDS analysis of the GPCR sequences from H. sapiens
# with projection of GPCRs from D. melanogaster
# as supplementary elements onto the space of human GPCRs
mmds_sup <- mmds.project(mmds_active, sup,system.file(
"csv/drome_gpcr_group.csv",package = "bios2mds"))

# displaying MDS coordinates
layout(matrix(1:6, 2, 3))

scree.plot(mmds_active$eigen.perc, lab = TRUE, title = "Scree plot of metric MDS")

mmds.2D <- mmds.2D.plot(mmds_active, title = "Sequence space of human GPCRs")

mmds.2D.plot(mmds_active,mmds_sup, title = "Projection of GPCRs from D. melanogaster
 onto the space space of human GPCRs ", active.alpha = 0.3)

# writing PDB files for 3D visualization of MDS coordinates
write.mmds.pdb(mmds_active,file.pdb=file)
```

`col.group` *Links elements in a mmds object to specific groups and colors*

Description

Links elements in a `mmds` object or `mmds.project` object to user-provided groups and colors.

Usage

```
col.group(x,file)
```

Arguments

`x`	a 'mmds' or 'project' object obtained from `mmds` or `mmds.project` function
`file`	a string of characters to indicate the file name assigning groups and colors to each active OR each supplementary element of the mmds object.

Details

`col.group` assigns each element of the mmds object to user-provided groupings and colors for coloring and labeling mmds scatter plots.

`col.group` requires a formatted file. See "csv/human_gpcr_group.csv" for an example. Each line corresponds to one element of the mmds object and must contain three parameters separated by ",".
The first parameter is the element name, as given in the multiple sequence alignment file.
The second parameter is the group name. Groupings must be provided by the user.
The third parameter is the group color in full letters (example : "black","green"). Two or more groups can have the same color, but elements within the same group must have the same color. The group is colored by the first color encountered.

Value

Adds data to a mmds object in order to color and label mmds scatter plots with user-provided groupings and colors.

Author(s)

Jean-Michel Becu

See Also

See `colors` function (default R package).
See `getcol` in `made4` package for special color palette developed to maximize the contrast between colors.

Examples

```
# performing metric MDS on human GPCRs with projection of
# GPCRs from D. melanogaster as supplementary data:
data(gpcr)
active <- gpcr$dif$sapiens.sapiens
mmds_active <- mmds(active)
mmds_active<-col.group(mmds_active,system.file("csv/human_gpcr_group.csv"
,package = "bios2mds"))
```

`dif` *Difference score*

Description

Measures the difference score between two aligned amino acid or nucleotide sequences.

Usage

```
dif(seq1, seq2, gap = FALSE, aa.strict = FALSE)
```

Arguments

`seq1`	a character vector representing a first sequence.
`seq2`	a character vector representing a second sequence.
`gap`	a boolean indicating whether the gap character should be taken as a supplementary symbol (TRUE) or not (FALSE). Default is FALSE.
`aa.strict`	a boolean indicating whether only strict amino acids should be taken into account (TRUE) or not (FALSE). Default is FALSE.

Details

The difference score between two aligned sequences is given by the proportion of sites that differs and is equivalent to $1 - PID$ (percent identity). `dif` is given by the number of aligned positions (sites) whose symbols differ, divided by the number of aligned positions. `dif` is equivalent to the *p* distance defined by Nei and Zhang (2006). In `dif`, positions with at least one gap can be excluded (gap = FALSE). When gaps are taken as a supplementary symbol (gap = TRUE), sites with gaps in both sequences are excluded.

From Nei and Zhang (2006), the *p* distance, which is the proportion of sites that differ between two sequences, is estimated by:

$$p = \frac{n_d}{n},$$

where *n* is the number of sites and n_d is the number of sites with different symbols.

The difference score ranges from 0, for identical sequences, to 1, for completely different sequences.

Value

A single numeric value representing the difference score.

Author(s)

Julien Pele

References

May AC (2004) Percent sequence identity: the need to be explicit. *Structure* **12**:737-738.

Nei M and Zhang J (2006) Evolutionary Distance: Estimation. *Encyclopedia of Life Sciences*.

Nei M and Kumar S (2000) Molecular Evolution and Phylogenetics. *Oxford University Press*, New York.

Examples

```
# calculating the difference score between the sequences
# of CLTR1_HUMAN and CLTR2_HUMAN:
aln <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
dif <- dif(aln$CLTR1_HUMAN, aln$CLTR2_HUMAN)
dif
```

---

dis *Dissimilarity score*

Description

Computes the dissimilarity score between two aligned amino acid sequences, based on substitution matrices.

Usage

```
dis(seq1, seq2, sub.mat.id = "PAM250", gap = NULL)
```

Arguments

seq1	a character vector representing a first amino acid sequence.
seq2	a character vector representing a second amino acid sequence.
sub.mat.id	a string of characters indicating the amino acid substitution matrix to be taken into account for calculation. This should be one of "PAM40", "PAM80", "PAM120", "PAM160", "PAM250", "BLOSUM30", "BLOSUM45", "BLOSUM62", "BLOSUM80", "GONNET", "JTT", "JTT_TM" and "PHAT". Default is PAM250. See sub.mat.
gap	a numeric vector of length 2, indicating the penalty for initiating and extending a strand of two gaps (gap ignored, default).

Details

Grishin and Grishin (2002) developped a method to calculate the similarity score with amino acid substitution matrices.

Let ***s*** be an amino acid substitution matrix with elements *s(a, b)*, let ***A*** be an alignment of *n* sequences, A_{ik} is a symbol (amino acid or gap: '-') in the site *k* of the sequence *i*. For each pair of sequences *i* and *j* from ***A***, the following equations**(1)**, **(2)**, **(3)** and **(4)** are calculated as follows:

(1) S_{ij}, called score per site, is obtained as:

$$S_{ij} = \sum_{k \in K_{ij}} s(A_{ik}, A_{jk})/l(K_{ij})$$

where K_{ik} is the set of sites *k* such that A_{ik} ! = '-' and A_{jk} ! = '-' and $l(K_{ij})$ is the number of elements in K_{ij}.

(2) T_{ij}, called average upper limit of the score per site, is obtained as:

$$T_{ij} = 0.5 \sum_{k \in K_{ij}} (s(A_{ik}, A_{ik}) + s(A_{jk}, A_{jk}))/l(K_{ij})$$

(3) S_{ij}^{rand}, called score per site expected from random sequences, is obtained as:

$$S_{ij}^{rand} = \sum_{a=1}^{20} \sum_{b=1}^{20} f_j^i(a) f_i^j(b) s(a, b)$$

where $f_j^i(a)$ is the frequency of amino acid 'a' in i^{th} protein sequence of ***A*** over all sites in K_{ij}.

(4) V_{ij}, called normalized score (Feng and Doolittle, 1997), is obtained as:

$$V_{ij} = \frac{S_{ij} - S_{ij}^{rand}}{T_{ij} - S_{ij}^{rand}}$$

The normalized score V_{ij} ranges from 0 (for random sequences) to 1 (for identical sequences). However, for very divergent sequences, V_{ij} can become negative due to statistical errors. In this case, `dis` attributes 0 to negative scores.

The dissimilarity score D_{ij} between sequences *i* and *j* is obtained from the similarity score as:

$$D_{ij} = V_{ij} - 1$$

Value

A single numeric value representing the dissimilarity score.

Author(s)

Julien Pele

References

Grishin VN and Grishin NV (2002) Euclidian space and grouping of biological objects. *Bioinformatics* **18**:1523-1534.

Feng DF and Doolittle RF (1997) Converting amino acid alignment scores into measures of evolutionary time: a simulation study of various relationships. *J Mol Evol* **44**:361-370.

Examples

```
# calculating dis between the sequences of CLTR1_HUMAN and CLTR2_HUMAN:
aln <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
dis <- dis(aln$CLTR1_HUMAN, aln$CLTR2_HUMAN)
dis
```

---

export.fasta *Writes a MSA file in FASTA format*

Description

Writes a multiple sequence alignment (MSA) file in FASTA format.

Usage

```
export.fasta(x, outfile = "alignment.fa", ncol = 60, open = "w")
```

Arguments

x	an object of class 'align', obtained from import.fasta or import.msf, or an element list from the extract.cluster function return.
outfile	a string of characters or string vector to indicate the name of the MSA file(s) to be written. If x is an object of class 'align', default is "alignment.fa". If x is an element list, for each element in x, default is the element name, followed by the ".alignement.fa" extension.
ncol	an integer value indicating the number of characters per line for the sequences in outfile.Default is 60.
open	a character value indicating the opening mode for outfile. This should be either of "w" to write into a new file, or "a" to append at the end of an already existing file. Default is "w".

Details

Initially, FASTA (for FAST-ALL) was the input format of the FASTA program, used for protein comparison and searching in databases. Presently, FASTA format is a standard format for biological sequences.

The FASTA formatted file of a single sequence displays:

- a single-line description beginning with a greater-than (>) symbol. The following word is the identifier.

- followed by any number of lines, representing biological sequence.

For multiple alignments, the FASTA formatted sequences are concatenated to create a multiple FASTA format.

Value

Produces a FASTA file for an 'align' object or a FASTA file for each cluster in list.

Note

For further information about FASTA format, see: https://www.ncbi.nlm.nih.gov/BLAST/fasta.shtml

Author(s)

Jean-Michel Becu

See Also

`write.fasta` function from `seqinr` package.

Examples

```
# reading of the multiple sequence alignment of human GPCRS in FASTA format:
wd <- tempdir()
#wd <- getwd()
file1 <- file.path(wd,"alignment.fa")
aln <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
export.fasta(aln,file1)
```

---

extract.cluster *Extraction of clusters alignments*

Description

Extracts the multiple sequence alignement of each cluster after K-means clustering.

Usage

```
extract.cluster(x, align)
```

Arguments

x	an object of class 'kmean', obtained from `kmeans.run` function.
align	an object of class 'align', obtained from `import.fasta` or `import.msf` function.

Details

Extraction of the MSA of each cluster.

Value

A named list of 'align' objects.

Author(s)

Jean-Michel Becu

Examples

```
# Clustering human GPCRs in 4 groups with 100 runs of K-means
# and extraction of the alignment of each cluster
aln <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
data(gpcr)
kmeans <- kmeans.run(gpcr$mmds$sapiens.active$coord, nb.clus = 4, nb.run = 100)
clusAlign <- extract.cluster(kmeans,aln)
```

---

gpcr *Pre-analyzed G-Protein-Coupled Receptor (GPCR) data set*

Description

This data set was obtained by the `bios2mds` analysis of the two multiple sequence alignment files provided in /msa. The MSA files were prepared as previously described by Deville et al. (2009) and updated with the July 2009 release of Uniprot (https://www.uniprot.org). They correspond to non-redundant sets of non-olfactory class A G-protein-coupled receptors (GPCRs) from *H. sapiens* and *D. melanogaster* (283 and 59 sequences, respectively). The data sets from *H. sapiens* and *D. melanogaster* constitute the active and supplementary data sets, respectively.

Usage

```
data(gpcr)
```

Format

`gpcr` is a named list of three elements:

dif a named list containing two distance matrices calculated from the `mat.dif` function (distances based on difference scores) with default parameters:

sapiens.sapiens a 283 by 283 matrix of difference scores between the 283 aligned sequences of human GPCRs.

melanogaster.sapiens a 59 by 283 matrix of difference scores between the aligned sequences of GPCRs from *H. sapiens* and *D. melanogaster* (283 and 59 sequences, respectively)..

dis a named list containing sixteen distance matrices calculated from the `mat.dis` function (distances based on dissimilarity scores) for the eight substitution matrices in `sub.mat` (other parameters by default):

sapiens.sapiens$* are 283 by 283 matrices of dissimilarity scores between the 283 aligned sequences of human GPCRs.

sapiens.sapiens$PAM40 is calculated with PAM40.

sapiens.sapiens$PAM80 is calculated with PAM80.

sapiens.sapiens$PAM120 is calculated with PAM120.

sapiens.sapiens$PAM250 is calculated with PAM250.

sapiens.sapiens$BLOSUM30 is calculated with BLOSUM30.

sapiens.sapiens$BLOSUM45 is calculated with BLOSUM45.

sapiens.sapiens$BLOSUM62 is calculated with BLOSUM62.

sapiens.sapiens$BLOSUM80 is calculated with BLOSUM80.

sapiens.sapiens$GONNET is calculated with GONNET.

sapiens.sapiens$JTT is calculated with JTT.

sapiens.sapiens$JTT_TM is calculated with JTT_TM.

sapiens.sapiens$PHAT is calculated with PHAT.

melanogaster.sapiens$* are 59 by 283 matrices of dissimilarity scores between the 283 aligned sequences from *H. sapiens* and *D. melanogaster*

melanogaster.sapiens$PAM40 is calculated with PAM40.

melanogaster.sapiens$PAM80 is calculated with PAM80.

melanogaster.sapiens$PAM120 is calculated with PAM120.

melanogaster.sapiens$PAM250 is calculated with PAM250.

melanogaster.sapiens$BLOSUM30 is calculated with BLOSUM30.

melanogaster.sapiens$BLOSUM45 is calculated with BLOSUM45.

melanogaster.sapiens$BLOSUM62 is calculated with BLOSUM62.

melanogaster.sapiens$BLOSUM80 is calculated with BLOSUM80.

melanogaster.sapiens$GONNET is calculated with GONNET.

melanogaster.sapiens$JTT is calculated with JTT.

melanogaster.sapiens$JTT_TM is calculated with JTT_TM.

melanogaster.sapiens$PHAT is calculated with PHAT.

mmds a typical example of metric MDS analysis with the `mmds` function of `bios2mds` using GPCRs from *H. sapiens* as active data and GPCRs from *D. melanogaster*, *N. vectensis* and *C. intestinalis* for supplementary data.

sapiens.active metric MDS analysis of active data from *H. sapiens*.

melanogaster.project projection of supplementary data from *D. melanogaster* onto the human active space.

vectensis.project projection of supplementary data from *N. vectensis* onto active the human space.

intestinalis.project projection of supplementary data from *C. intestinalis* onto the human active space.

Source

Deville J, Rey J and Chabbert M (2009) An indel in transmembrane helix 2 helps to trace the molecular evolution of class A G-protein-coupled receptors. *J Mol Evol* **68**: 475- 489.

Examples

```
# loading gpcr
data(gpcr)

# displaying the matrix of differences scores between GPCRs sequences
# from H. sapiens
gpcr$dif$sapiens.sapiens

# displaying the matrix of dissimilarity scores between GPCRs sequences
# from H. sapiens
gpcr$dis$sapiens.sapiens$PAM250

# displaying the matrix of dissimilarity scores between GPCRs sequences
# from H. sapiens and D. Melanogaster calculated with the BLOSUM45 matrix
gpcr$dis$melanogaster.sapiens$BLOSUM45

# displaying mmds analysis of the MSA of GPCRs from H. sapiens
# and D. Melanogaster
gpcr$mmds$sapiens.active
gpcr$mmds$melanogaster.project
```

`import.fasta` *Reads a file in FASTA format*

Description

Reads a Multiple Sequence Alignment (MSA) file in FASTA format (.fasta or .fa extension).

Usage

```
import.fasta(file, aa.to.upper = TRUE, gap.to.dash = TRUE)
```

Arguments

`file`	a string of characters to indicate the name of the MSA file to be read.
`aa.to.upper`	a logical value indicating whether amino acids should be converted to upper case (TRUE) or not (FALSE). Default is TRUE.
`gap.to.dash`	a logical value indicating whether the dot (.) and tilde ($\sim$) gap symbols should be converted to dash (-) character (TRUE) or not (FALSE). Default is TRUE.

Details

Initially, FASTA (for FAST-ALL) was the input format of the FASTA program, used for protein comparison and searching in databases. Presently, FASTA format is a standard format for biological sequences.

The FASTA formatted file of a single sequence displays:

- a single-line description beginning with a greater-than (>) symbol. The following word is the identifier.
- followed by any number of lines, representing biological sequence.

For multiple alignments, the FASTA formatted sequences are concatenated to create a multiple FASTA format.

Value

A object of class 'align', which is a named list whose elements correspond to sequences, in the form of character vectors.

Note

For further information about FASTA format, see: https://www.ncbi.nlm.nih.gov/BLAST/fasta.shtml

Author(s)

Julien Pele

References

Pearson WR and Lipman DJ (1988) Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A* **27**:2444-2448.

See Also

`read.fasta` function from `bio3d` package.
`read.fasta` function from `seqinr` package.
`read.FASTA` function from `aaMI` package (archived).

Examples

```
# reading of the multiple sequence alignment of human GPCRS in FASTA format:
aln <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
```

`import.msf` *Reads a multiple sequence alignment file in MSF format*

Description

Reads a Multiple Sequence Alignment (MSA) file in MSF format (.msf extension).

Usage

```
import.msf(file, aa.to.upper = TRUE, gap.to.dash = TRUE)
```

Arguments

`file`	a string of characters to indicate the name of the MSA file to be read.
`aa.to.upper`	a logical value indicating whether amino acids should be converted to upper case (TRUE) or not (FALSE). Default is TRUE.
`gap.to.dash`	a logical value indicating whether the dot (.) and tilde ($\sim$) gap symbols should be converted to the dash (-) character (TRUE) or not (FALSE). Default is TRUE.

Details

Initially, Multiple Sequence Format (MSF) was the multiple sequence alignment format of the Wisconsin Package (WP) or GCG (Genetic Computer Group). This package is a suite of over 130 sequence analysis programs for database searching, secondary structure prediction or sequence alignment. Presently, numerous multiple sequence alignment editors (Jalview and GeneDoc for example) can read and write MSF files.

MSF file displays several specificities:

- a header containing sequence identifiers and characteristics (length, check and weight).
- a separator symbolized by 2 slashes (//).
- sequences of identifiers, displayed by consecutive blocks.

Value

A object of class 'align', which is a named list whose elements correspond to sequences, in the form of character vectors.

Note

`import.msf` checks the presence of duplicated identifiers in header. Sequences whose identifiers are missing in header are ignored.

Author(s)

Julien Pele

See Also

`read.alignment` function from `seqinr` package.
`read.GDoc` function from `aaMI` package (archived).

Examples

```
# reading of the multiple sequence alignment of human GPCRs in MSF format:
aln <- import.msf(system.file("msa/human_gpcr.msf", package = "bios2mds"))
```

`kmeans.run` *Multiple runs of K-means analysis*

Description

Performs multiple runs of K-means clustering and analyzes data.

Usage

```
kmeans.run(mat, nb.clus = 2, nb.run = 1000, iter.max = 10000,
method = "euclidean")
```

Arguments

`mat`	a numeric matrix representing the coordinates of the elements after metric MDS analysis.
`nb.clus`	a numeric value indicating the number of clusters. Default is 2.
`nb.run`	a numeric value indicating the number of runs. Default is 1000.
`iter.max`	a numeric value indicating the maximum number of iterations for K-means. Default is 10000.
`method`	a string of characters to determine the distance to be used. This should be one of "euclidean", "maximum", "manhattan", "canberra", "binary", "pearson", "correlation", "spearman" or "kendall". Default is "euclidean".

Details

The aim of K-means clustering is the partition of elements into a user-provided number of clusters. Several runs of K-means analysis on the same data may return different cluster assignments because the K-means procedure attributes random initial centroids for each run. The robustness of an assignment depends on its reproducibility.

The function `matchClasses` from the `e1071` package is used to compare the cluster assignments of the different runs and returns a score of agreement between them. The most frequent clustering solution is selected and used as a reference to assess the reproducibility of the analysis.

`kmeans.run` returns two lists. In either list, the clusters refer to those observed in the most frequent solution. The first list provides, for each element, the relative ratio of its assignment to each cluster in the different runs. The second list provides, for each cluster, the list of the assigned elements along with the relative assignment to this cluster in the different runs.

Value

A object of class 'kmean', which is a named list of two elements

elements	a named list of elements with the relative assignment of each element to each cluster.
clusters	a named list of clusters with the elements assigned to this cluster in the most frequent solution and their relative assignment to this cluster in multiple runs.

Note

During the K-means procedure, an empty cluster can be obtained if no objects are allocated to the cluster. In kmeans.run, runs with empty clusters are discarded.

kmeans.run requires Kmeans and matchClasses functions from amap and e1071 packages, respectively.

Author(s)

Julien Pele

Examples

```
# Clustering human GPCRs in 4 groups with 100 runs of K-means
data(gpcr)
coord <- gpcr$mmds$sapiens.active$coord
kmeans.run1 <- kmeans.run(coord, nb.clus = 4, nb.run = 100)
kmeans.run1$clusters
kmeans.run1$elements
```

---

mat.dif *Matrices of difference scores between sequences*

Description

Computes a matrix providing the distances based on the difference scores between sequences from two multiple sequence alignments.

Usage

```
mat.dif(align1, align2, gap = FALSE, aa.strict = FALSE, sqrt = FALSE)
```

Arguments

align1	a list of character vectors representing a first multiple sequence aligment.
align2	a list of character vectors representing a second multiple sequence aligment.
gap	a logical value indicating whether gap character should be taken as supplementary symbol (TRUE) or not (FALSE). Default is FALSE.

aa.strict	a logical value indicating whether only strict amino acids should be taken into account (TRUE) or not (FALSE). To be used only for amino acid sequences. Default is FALSE.
sqrt	a logical value indicating whether the distance should be equal to the squared root of the difference score (TRUE) or not (FALSE). Default is FALSE.

Details

If align1 and align2 are identical, mat.dif computes the symetrical matrix of distances between each sequence of the alignment.

Before using mat.dif, users must check the alignment of sequences within align1 and align2 and between align1 and align2.

Value

A named numeric matrix providing the difference-based distances between each pair of sequences from align1 and align2. The number of rows and columns is identical to the number of sequences in align1 and align2, respectively.

Author(s)

Julien Pele and Jean-Michel Becu

See Also

identity function from bio3d package.

Examples

```
# calculating the matrix of distances based on the difference scores
# between GPCRs sample from H. sapiens and D. melanogaster:
aln_human <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
aln_drome <- import.fasta(system.file("msa/drome_gpcr.fa", package = "bios2mds"))
mat.dif1 <- mat.dif(aln_human[1:5], aln_drome[1:5])
mat.dif1
```

---

mat.dis *Matrices of dissimilarity scores between amino acid sequences*

Description

Computes a matrix providing the distances based on dissimilarity scores between sequences from two multiple sequence alignments.

Usage

```
mat.dis(align1, align2, sub.mat.id = "PAM250", sqrt=FALSE)
```

Arguments

align1	a list of character vectors representing a first multiple sequence aligment.
align2	a list of character vectors representing a second multiple sequence aligment.
sub.mat.id	a string of characters indicating the amino acid substitution matrix used for calculation of the dissimilarity score. This should be one of "PAM40", "PAM80", "PAM120", "PAM160", "PAM250", "BLOSUM30", "BLOSUM45", "BLOSUM62", "BLOSUM80", "GONNET", "JTT", "JTT_TM" and "PHAT". The supported substitution matrices are in sub.mat. Default is PAM250.
sqrt	a logical value indicating whether the distance should be equal to the squared root of the difference score (TRUE) or not (FALSE). Default is FALSE.

Details

The dissimilarity score between a sequence *i* from align1 and a sequence *j* from align2 is calculated with an amino acid substitution matrix from sub.mat.

If align1 and align2 are identical, mat.dis computes the symetrical matrix of distances between each sequence of the alignment.

Before using mat.dis, users must check the alignment of sequences within align1 and align2 and between align1 and align2.

Value

A named numeric matrix providing the dissimilarity-based distances between each pair of sequences from align1 and align2, based on the substitution matrix sub.mat.id. The number of rows and columns is identical to the number of sequences in align1 and align2, respectively.

Author(s)

Julien Pele and Jean-Michel Becu

Examples

```
# calculating dissimilarity distances between GPCR sequences sample from
#H. sapiens and D. melanogaster, based on the PAM250 matrix:
aln_human <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
aln_drome <- import.fasta(system.file("msa/drome_gpcr.fa", package = "bios2mds"))
mat.dis1 <- mat.dis(aln_human[1:5], aln_drome[1:5])
mat.dis1

# calculating dissimilarity distances between GPCRs sequences sample from
#H. sapiens and D. melanogaster, based on the BLOSUM45 matrix:
aln_human <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
aln_drome <- import.fasta(system.file("msa/drome_gpcr.fa", package = "bios2mds"))
mat.dis1 <- mat.dis(aln_human[1:5], aln_drome[1:5], sub.mat.id = "BLOSUM45")
mat.dis1
```

`mmds` *Metric multidimensional scaling*

Description

Performs metric MultiDimensional Scaling (MDS) analysis of active elements.

Usage

```
mmds(active, pc = 3, group.file = NULL)
```

Arguments

`active`	a numeric matrix of distances between active elements.
`pc`	a numeric value indicating the number of principal components to be saved. Default is 3.
`group.file`	a string of characters to indicate the file name assigning groups and colors to each active element.

Details

Metric multidimensional scaling is a statistical analysis technique aimed at analyzing a matrix of distances between 'active' elements. MDS maps the elements onto a low dimensional space (usually 2D or 3D), In this space, elements are represented by points whose respective distances best approximate the initial distance.

`active` must have some characteristics:

- `active` represents the matrix of pairwise distances between active elements. The active matrix must be symmetric (square and equals to its transpose). The distances on the main diagonal must be equal to 0. The distances do not have to be Euclidean. They can just express a difference or dissimilarity score, with a 0 value between same elements and a positive value between different elements.

The method for the computation of metric MDS is described by Abdi (2007). Briefly, if *N* is the number of active sequences and *D* is the *N* by *N* matrix of the squared distances computed from the active matrix, the `mmds` function performs the following steps:

(1) Transforms **D** into a cross-product matrix **S**:

$$\boldsymbol{S} = -0.5[\boldsymbol{I} - \boldsymbol{m}^T\mathbf{1}] \times \boldsymbol{D} \times [\boldsymbol{I} - \boldsymbol{m}^T\mathbf{1}]^T$$

where **I** is the *N* by *N* identity matrix, **m** is the *N* by *1* matrix mass, where each mass equal to $\frac{1}{N}$ and **1** is an *N* by *N* matrix of ones.

(2) Transforms ***S*** into a factor matrix ***F***:

$$\boldsymbol{F} = \boldsymbol{M}^{\frac{-1}{2}}\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$$

where **M** is the diag{**m**}, $\boldsymbol{U}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues, such as $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\boldsymbol{T}}$, where $^{\boldsymbol{T}}$ denotes the transposition operation.

The eigenvectors of $\boldsymbol{S}$, also called principal components (whose number is smaller or equal to $\boldsymbol{N}$), form the active space. $\boldsymbol{F}$ gives the coordinates of the active elements in this space.

Value

A object of class 'mmds', which is a named list of five elements:

`eigen`	a numeric vector of the eigenvalues.
`eigen.perc`	a numeric vector of the relative eigenvalues (eigenvalues divided by the sum of the absolute eigenvalues).
`coord`	a named numeric matrix representing the coordinates of active elements.
`group`	a named string matrix representing the differents groups of elements and the associate color (Default is 'NoGroup' and 'black').
`col`	a named string matrix representing, foreach named elements, the associate group and color (Default is 'NoGroup' and 'black').
`source`	a named list with 2 elements, the matrix (**D**) of squared distance and the vector of mass **m**, that will used for projection of supplementary elements, if required, with the `mmds.project` function.

Note

If `active` do not contain names:
A tag "A" followed by an incremented number names the rows and the columns of `active`.

Author(s)

Julien Pele and Jean-Michel Becu

References

Abdi H (2007) Metric multidimensional scaling. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics.* Thousand Oaks (CA): Sage. pp. 598-605.

For further information on multidimentional scaling:

Takane Y, Jung S and Oshima-Takane Y (2009) Multidimensional scaling, in *Handbook of quantitative methods in psychology*, eds Millsap R, Maydeu-Olivares A (Sage Publications, London) pp. 219-242.

Borg I and Groenen PJF (2005) Modern multidimensional scaling. New York : Springer.

Gower JC (1967) A comparison of some methods of cluster analysis. *Biometrics* **23**:623-637.

ToRgerson WS (1958) Theory and methods of scaling. New York : Wiley.

See Also

`cmdscale` function from `stats` package.
`dudi.pco`, `suprow` and `supcol` functions from `ade4` package.
`PCA` function from `FactoMineR` package.

Examples

```
# performing metric MDS of human GPCRs with projection of
# GPCRs from D. melanogaster as supplementary elements:
data(gpcr)
active <- gpcr$dif$sapiens.sapiens
mmds1 <- mmds(active = active)
mmds1$active.coord
```

`mmds.2D.multi` *Plots the mmds coordinates onto a 2D space*

Description

Displays a scatter plot of the active elements and the barycenter of supplementary elements, or of groups of supplementary elements, after a metric MDS analysis.

Usage

```
mmds.2D.multi(x,project, title = NULL, axis = c(1, 2), xlim = NULL,
ylim = NULL, outfile.type = NULL,bary="p",
outfile.name = "mmds",new.plot = TRUE, active.col = x$col[,3],
active.alpha = 0.6, active.pch = 20, sup.pch = NULL , active.cex = 2,
sup.cex = 2, active.legend.cex = 2, sup.legend.cex = 2,
active.legend.lwd = 1, sup.legend.lwd = 2, active.lwd = 1, sup.lwd = 3,
legend = TRUE, active.legend.pos = "bottomleft",
sup.legend.pos = "bottomright", group.name = NULL,
ensemble.legend.name = "", group.col = NULL, outfile.width = NULL, outfile.height = NULL,
box.lwd = 1, cex.axis = 1, sup.legend.text = 1,
active.legend.text = 1, legend.axis = TRUE, grid = TRUE, axes = TRUE)
```

Arguments

`x`	an object of class 'mmds', obtained from `mmds` function.
`project`	an object of class 'project' or a list of objects of class 'project', obtained from the `mmds.project` function. In the list the order of 'project' objects is very important because if the `bary` option equal to "l", the barycenters are linked in the same order as the list with the active barycenter linked as the last object.
`title`	a string of characters representing the title of the plot. Default is "Metric MDS".
`bary`	a character to specify the representation of group barycenter with "p" for representation of group barycenter foreach 'project' and active space by dots and "l" for representation of barycenter linked foreach group in order of appearence in project list and with the barycenter of active group in last position. Default is "p".
`axis`	a numeric vector of length two representing the principal components displayed on the plot. Default is c(1, 2).
`xlim`	a numeric vector representing the range for the x values. Default is full range.

`ylim` a numeric vector representing the range for the y values. Default is full range.

`outfile.type` a string indicating the extension type of the graph outfile. Default is NULL. If not NULL, this should be one of "pdf", "tiff", "png" or "postscript". In this case, the parameter outfile.name and the forthcoming parameters are activated.

`outfile.name` a string ("mmds", default) indicating the name and directory of pdf graph outfile. The extension file is added automaticaly. See `outfile.type`.

`new.plot` a boolean indicating whether a new graphical device should be created (TRUE) or not (FALSE). Default is TRUE.

`active.pch` an integer indicating the symbol of active elements. Default is 20, corresponding to dots.

`sup.pch` an integer vector representing the symbol for project barycenter. Default is NULL. If sup.pch is NULL then symbol are choose randomly.

`active.col` a string of characters or character vector representing the color(s) of the active elements. Default is x$col[,3]. It corresponds either to the user-provided colors if the `col.group` function has been used previously, or to black (filled automaticaly with the `mmds` function).

`active.alpha` a numeric value indicating the alpha value for opacity of active objects. This value must range from 0 (invisible) to 1 (full opacity). Default is 0.6.

`active.cex` a numeric value indicating the size of the active symbols. Default is 2.

`sup.cex` a numeric value indicating the size of the supplementary symbols. Default is 2.

`active.legend.cex` a numeric value indicating the size of active symbols in legend. Default is 2.

`sup.legend.cex` a numeric value indicating the size of supplementary symbols in legend. Default is 2.

`active.lwd` a numeric value indicating the width of active symbols. Default is 1.

`sup.lwd` a numeric value indicating the width of supplementary symbols. Default is 3.

`active.legend.lwd` a numeric value indicating the width of active symbols in legend. Default is 1.

`sup.legend.lwd` a numeric value indicating the width of supplementary objects in legend. Default is 4.

`legend` a boolean indicating whether the legend should be displayed (TRUE) or not (FALSE). Default is TRUE.

`active.legend.pos` a string indicating the position of the legend for active elements. Default is "topleft".

`sup.legend.pos` a string indicating the position of the legend for supplementary elements. Default is "topright".

`group.name` a string vector indicating the names of the used groups. NULL vector allow to use all groups present in the 'mmds' object (x) and in all 'mmds.project' object. It corresponds either to the user-provided groups if the `col.group` function has been used previously, or to "NoGroup" (filled automaticaly with the `mmds` ans `mmds.project` functions). Each group in this vector must be presents in 'mmds' or 'mmds.project' objects and only group present in this vector will draw in the plot. Default is NULL.

`ensemble.legend.name`
: a string vector indicating the names of the `project` and the name of active space. Default is "". It corresponds either to the names of project object ans in the last position of vector the name of the active space.

`group.col`
: a string vector indicating the colors of appearance of the used groups. This vector must have the same length of the group.name vector. NULL vector associate automaticaly the color of groups in 'mmds' and 'mmds.project' object to the group in the group.name parameter. It corresponds either to the user-provided colors if the `col.group` function has been used previously, or to black (filled automaticaly with the `mmds` and `project.mmds` functions). Default is NULL.

`outfile.width`
: a numeric value in inches indicating the width of graph outfile. Default differs by outfile.type. See `pdf`, `png`, `postscript`. The resolution for tiff and png figures is 150 dpi.

`outfile.height`
: a numeric value in inches indicating the height of graph outfile. Default differs by outfile.type. The resolution for tiff and png figures is 150 dpi. See `pdf`, `tiff`, `png`, `postscript`.

`box.lwd`
: a numeric value indicating the border width of graph box and legend box. Default is 1.

`cex.axis`
: a numeric value indicating the character size for the x and y axes. Default is 1.

`sup.legend.text`
: a numeric value indicating the character size of the supplementary tag in legend. Default is 1.

`active.legend.text`
: a numeric value indicating the character size of active tag in legend. Default is 1.

`legend.axis`
: a boolean indicating whether axis name should be displayed (TRUE) or not (FALSE). Default is TRUE.

`grid`
: a boolean indicating whether grid should be displayed (TRUE) or not (FALSE). Default is TRUE.

`axes`
: a boolean indicating whether x and y axes should be displayed (TRUE) or not (FALSE). Default is TRUE.

Details

If `mmds.2D.plot` is used after the `col.group` function, the elements are colored by the color scheme provided in the .csv file (see `col.group` for details). If the `col.group` function has not been used, the default colors are black and magenta for active and supplementary elements.

`mmds.2D.plot` helps identify patterns in data and compare active and supplementary elements.

`active.alpha` argument is helpful for visualization of supplementary elements because it allows the symbols of supplementary elements to be in the foreground as compared to active elements.

Value

Produces a scatter plot on the active graphical device.

Note

mmds.2D.multi requires alpha function from scales package.

Author(s)

Jean-Michel Becu

See Also

plot.PCA function from FactoMineR package.
png, pdf, postscript functions (default R package).

Examples

```
# scatter plot of human GPCRs onto the first two axes obtained
# from MDS analysis with projection of GPCRs from N.vectensis
# and C.intestinalis as supplementary elements:
data(gpcr)

mmds_human <- gpcr$mmds$sapiens.active
project_vectensis <-gpcr$mmds$vectensis.project
project_intestinalis <-gpcr$mmds$intestinalis.project

mmds.2D.multi(mmds_human,project=list(project_vectensis,project_intestinalis),
bary='l',cex.axis=0.01,active.cex = 1, sup.cex = 1,active.lwd=1.5,sup.lwd=3,
ensemble.legend.name=c('nemve','inte','human'),legend=FALSE,title='multi_human')

# with selected group

mmds.2D.multi(mmds_human,project=list(project_vectensis,project_intestinalis),
bary='l',cex.axis=0.01,active.cex = 1, sup.cex = 1,active.lwd=1.5,sup.lwd=3,
ensemble.legend.name=c('nemve','inte','human'),legend=FALSE,title='multi_human',
group.name=c('SO','PEP','OPN','ADENO'),group.col=c("red","forestgreen","orange","maroon"))

# a single projection

mmds.2D.multi(mmds_human,project=project_vectensis,
bary='l',cex.axis=0.01,active.cex = 1, sup.cex = 1,active.lwd=1.5,sup.lwd=3,
ensemble.legend.name=c('nemve','human'),legend=FALSE,title='nemve_human')
```

---

mmds.2D.plot *Plots the mmds coordinates onto a 2D space*

Description

Displays a scatter plot of the active elements and, if present, of supplementary elements, after a metric MDS analysis.

Usage

```
mmds.2D.plot(x,project = NULL, title = NULL, axis = c(1, 2), xlim = NULL,
ylim = NULL, outfile.type = NULL, outfile.name = "mmds",new.plot = TRUE,
active.col = x$col[,3], active.alpha = 1, sup.col = project$col[,3],
active.pch = 20, sup.pch = 3, active.lab = FALSE, sup.lab = FALSE,
active.cex = 2, sup.cex = 2, active.legend.cex = 2, sup.legend.cex = 2,
active.legend.lwd = 1, sup.legend.lwd = 2, active.lwd = 1, sup.lwd = 4,
legend = TRUE, active.legend.pos = "bottomleft",
sup.legend.pos = "bottomright", active.legend.name = x$group[,1],
sup.legend.name = project$group[,1], active.legend.col = x$group[,2],
sup.legend.col = project$group[,2], outfile.width = NULL, outfile.height = NULL,
box.lwd = 1, cex.axis = 1, cex.lab = 1, sup.legend.text = 1,
active.legend.text = 1, legend.axis = TRUE, grid = TRUE, axes = TRUE)
```

Arguments

x	an object of class 'mmds', obtained from mmds function.
project	an object of class 'project', obtained from mmds.project function.
title	a string of characters representing the title of the plot. Default is "Metric MDS".
axis	a numeric vector of length two representing the principal components displayed on the plot. Default is c(1, 2).
xlim	a numeric vector representing the range for the x values. Default is full range.
ylim	a numeric vector representing the range for the y values. Default is full range.
outfile.type	a string indicating the extension type of the graph outfile. Default is NULL. If not NULL, this should be one of "pdf", "tiff", "png" or "postscript". In this case, the parameter outfile.name and the forthcoming parameters are activated.
outfile.name	a string ("mmds", default) indicating the name and directory of pdf graph outfile. The extension file is added automaticaly. See outfile.type.
new.plot	a boolean indicating whether a new graphical device should be created (TRUE) or not (FALSE). Default is TRUE.
active.pch	an integer indicating the symbol of active elements. Default is 20, corresponding to dots.
sup.pch	an integer indicating the symbol of supplementary elements. Default is 3, corresponding to crosses.
active.col	a string of characters or character vector representing the color(s) of the active elements. Default is x$col[,3]. It corresponds either to the user-provided colors if the col.group function has been used previously, or to black (filled automaticaly with the mmds function).
active.alpha	a numeric value indicating the alpha value for opacity of active objects. This value must range from 0 (invisible) to 1 (full opacity). Default is 1.
sup.col	a string or character vector representing the color(s) of supplementary elements. Default is project$col[,3]. It corresponds either to the user-provided colors if the col.group function has been used previously, or to magenta (filled automaticaly with the mmds function).

`active.lab` a boolean indicating whether labels of active elements should be displayed (TRUE) or not (FALSE). Default is FALSE.

`sup.lab` a boolean indicating whether labels of supplementary elements should be displayed (TRUE) or not (FALSE). Default is FALSE.

`active.cex` a numeric value indicating the size of the active symbols. Default is 2.

`sup.cex` a numeric value indicating the size of the supplementary symbols. Default is 2.

`active.legend.cex`
a numeric value indicating the size of active symbols in legend. Default is 2.

`sup.legend.cex` a numeric value indicating the size of supplementary symbols in legend. Default is 2.

`active.lwd` a numeric value indicating the width of active symbols. Default is 1.

`sup.lwd` a numeric value indicating the width of supplementary symbols. Default is 4.

`active.legend.lwd`
a numeric value indicating the width of active symbols in legend. Default is 1.

`sup.legend.lwd` a numeric value indicating the width of supplementary objects in legend. Default is 4.

`legend` a boolean indicating whether the legend should be displayed (TRUE) or not (FALSE). Default is TRUE.

`active.legend.pos`
a string indicating the position of the legend for active elements. Default is "topleft".

`sup.legend.pos` a string indicating the position of the legend for supplementary elements. Default is "topright".

`active.legend.name`
a string vector indicating the names of the `active` groups. Default is x$group[,1]. It corresponds either to the user-provided groups if the `col.group` function has been used previously, or to "NoGroup" (filled automaticaly with the `mmds` function).

`sup.legend.name`
a string vector indicating the names of the `sup` groups. Default is project$group[,1]. It corresponds either to the user-provided groups if the `col.group` function has been used previously, or to "NoGroup" (filled automaticaly with the `mmds` function).

`active.legend.col`
a string vector indicating the colors of the different `active` groups. Default is x$group[,2]. It corresponds either to the user-provided colors if the `col.group` function has been used previously, or to black (filled automaticaly with the `mmds` function).

`sup.legend.col` a string vector indicating the colors of the different `sup` groups. Default is project$group[,2]. It corresponds either to the user-provided colors if the `col.group` function has been used previously, or to magenta (filled automaticaly with the `mmds` function).

`outfile.width` a numeric value in inches indicating the width of graph outfile. Default differs by outfile.type. See `pdf`, `png`, `postscript`. The resolution for tiff and png figures is 150 dpi.

`outfile.height` a numeric value in inches indicating the height of graph outfile. Default differs by outfile.type. The resolution for tiff and png figures is 150 dpi. See `pdf`, `tiff`, `png`, `postscript`.

`box.lwd` a numeric value indicating the border width of graph box and legend box. Default is 1.

`cex.axis` a numeric value indicating the character size for the x and y axes. Default is 1.

`cex.lab` a numeric value indicating the character size for the labels of the x and y axes. Default is 1.

`sup.legend.text` a numeric value indicating the character size of the supplementary tag in legend. Default is 1.

`active.legend.text` a numeric value indicating the character size of active tag in legend. Default is 1.

`legend.axis` a boolean indicating whether axis name should be displayed (TRUE) or not (FALSE). Default is TRUE.

`grid` a boolean indicating whether grid should be displayed (TRUE) or not (FALSE). Default is TRUE.

`axes` a boolean indicating whether x and y axes should be displayed (TRUE) or not (FALSE). Default is TRUE.

Details

If `mmds.2D.plot` is used after the `col.group` function, the elements are colored by the color scheme provided in the .csv file (see `col.group` for details). If the `col.group` function has not been used, the default colors are black and magenta for active and supplementary elements.

`mmds.2D.plot` helps identify patterns in data and compare active and supplementary elements.

`active.alpha` argument is helpful for visualization of supplementary elements because it allows the symbols of supplementary elements to be in the foreground as compared to active elements.

Value

Produces a scatter plot on the active graphical device.

Note

`mmds.2D.plot` requires `alpha` function from `scales` package.

Author(s)

Julien Pele and Jean-Michel Becu

See Also

`plot.PCA` function from `FactoMineR` package.
`png`, `pdf`, `postscript` functions (default R package).

Examples

```
# scatter plot of human GPCRs onto the first two axes obtained from MDS analysis
# with projection of GPCRs from D. melanogaster as supplementary elements:
data(gpcr)
active <- gpcr$dif$sapiens.sapiens
mmds_active <- mmds(active,group.file=system.file(
"csv/human_gpcr_group.csv",package = "bios2mds"))
mmds.2D.plot(mmds_active, active.alpha = 0.5, active.lab = TRUE)

# with group information
sup <- gpcr$dif$melanogaster.sapiens
mmds_sup <- mmds.project(mmds_active, sup,group.file=system.file(
"csv/drome_gpcr_group.csv",package = "bios2mds"))
mmds.2D.plot(mmds_active,mmds_sup)
```

---

mmds.3D.plot *Display the mmds coordinates onto a 3D space*

Description

Displays a 3D plot of the active elements and, if present, of supplementary elements, after a metric MDS analysis.

Usage

```
mmds.3D.plot(x, project = NULL, title = NULL, axis = c(1:3),
active.type = "s", sup.type = "p", active.size = 2, radius = 0.005,
sup.size = 10, active.col = x$col[,3], sup.col = project$col[,3],
box = TRUE, axes = TRUE, new.plot = TRUE, label = TRUE,
xlim = NULL, ylim = NULL, zlim = NULL, box.lwd = 2,
box.antialias = TRUE, ...)
```

Arguments

x	an object of class 'mmds', obtained from mmds function.
title	a string of characters representing the title of the plot. Default is "Metric MDS".
project	an object of class 'project', obtained from mmds.project function, or a list of object of class 'project'.
axis	a numeric vector of length three representing the principal components displayed on the plot. Default is c(1:3).
active.type	an character indicating the symbol of active elements. This should be one of "s" for spheres, "p" for points, "l" for lines, "h" for line segments from z=0 and "n" for none. Default is "s", corresponding to spheres.
sup.type	an character indicating the symbol of supplementary elements. This should be one of "s" for spheres, "p" for points, "l" for lines, "h" for line segments from z=0 and "n" for none. Default is "p", corresponding to points.

active.col	a string of characters or character vector representing the color(s) of the active elements. Default is x$col[,3]. It corresponds either to the user-provided colors if the col.group function has been used previously, or to black (filled automaticaly with the mmds function).
sup.col	a string or character vector representing the color(s) of supplementary elements. Default is project$col[,3]. It corresponds either to the user-provided colors if the col.group function has been used previously, or to magenta (filled automaticaly with the mmds function)
active.size	a numeric value indicating the size of active symbols. Default is 2.
sup.size	a numeric value indicating the size of supplementary symbols. Default is 20.
box	a boolean indicating whether the box should be displayed (TRUE) or not (FALSE). Default is TRUE.
axes	a boolean indicating whether axes should be displayed (TRUE) or not (FALSE). Default is TRUE.
radius	a numeric value indicating the radius of spheres symbols only. If x.type equal to "s" the x.size parameter was inactivated. Default is 0.01.
new.plot	a boolean indicating whether a new 3D plot create/replace active 3D device (TRUE) or not to insert in it (FALSE). Default is TRUE.
label	a boolean indicating whether the label axes should be displayed (TRUE) or not (FALSE). Default is TRUE.
xlim	a numeric vector representing the range for the x values. Default is full range.
ylim	a numeric vector representing the range for the y values. Default is full range.
zlim	a numeric vector representing the range for the z values. Default is full range.
box.lwd	a numeric value indicating the width of box and axes lines. Default is 2.
box.antialias	a boolean specifying if box and axes lines should be antiliased. Default is TRUE.
...	additional parameters which will be passed to par3d, material3d and decorate3d of rgl package.

Details

If mmds.3D.plot is used after the col.group function, the elements are colored by the color scheme provided in the .csv file (see col.group for details). If the col.group function has not been used, the default colors are black and magenta for active and supplementary elements.

mmds.3D.plot helps identify patterns in data and compare active and supplementary elements.

Value

Produces a 3D plot on graphical device.

Note

mmds.3D.plot requires plot3D function from rgl package. See rgl documentation to supplementary function like snapshot3D to save image file of 3D device.

Author(s)

Jean-Michel Becu

Examples

```
# 3D plot of human GPCRs onto the first three axes obained from MDS analysis
# with projection of GPCRs from D. melanogaster as supplementary elements:
data(gpcr)
mmds.3D.plot(gpcr$mmds$sapiens.active,active.type="p",label=FALSE,lit=FALSE,
point_antialias=TRUE,box.lwd=3,sup.size=4.3,active.size=4.3)
bbox3d(shininess=0.5)
```

---

mmds.plot *Plots a summary of the mmds results*

Description

Displays one scree plot and three scatter plots of mmds results.

Usage

```
mmds.plot(x, project = NULL, new.plot = TRUE, pdf.file = NULL)
```

Arguments

x	an object of class 'mmds', obtained from mmds.
project	an object of class 'project', obtained from mmds.project function.
new.plot	a boolean indicating whether a new graphical device should be created (TRUE) or not (FALSE). Default is TRUE.
pdf.file	a string indicating the name and directory of the pdf graph outfile. Default is NULL. If this parameter is not NULL, the parameter new.plot is inactivated.

Details

mmds.plot is a wrapper calling of both scree.plot and mmds.2D.plot. It produces a 2x2 plot with one scree plot of the relative eigenvalues, in the upper left, and three scatter plots. The three scatter plots are generated as follows:

- scatter plot of the elements on the first and second components in the upper right.
- scatter plot of the elements on the first and third components in the lower left.
- scatter plot of the elements on the second and third components in the lower right.

If object x contains supplementary elements, they are also projected onto the three scatter plots. The active and supplementary elements are represented by dots and crosses, respectively. The color.group function may be used before calling mmds.plot to color elements by user-provided groups.

Value

Produces a summary plot of the MDS analysis on the same active graphical device.

Note

The scatter plots can display supplementary objects if their coordinates are present in x input.

Author(s)

Julien Pele and Jean-Michel Becu

See Also

plot.pca function in bio3d package.

Examples

```
# summary plot of the MDS analysis of human GPCRs with projection of GPCRs
# from D. melanogaster as supplementary elements:
data(gpcr)
mmds.plot(gpcr$mmds$sapiens.active,gpcr$mmds$melanogaster.project)
```

---

mmds.project *Metric multidimensional Scaling Projection*

Description

Performs metric MultiDimensional Scaling (MDS) analysis of active elements and projects supplementary elements onto the active space defined by active elements.

Usage

```
mmds.project(mmds, sup, pc = 3, group.file = NULL)
```

Arguments

mmds	an object of class 'mmds', obtained from mmds, providing the active elements.
sup	a numeric matrix of distances between supplementary and active elements.
pc	a numeric value indicating the number of principal components to be saved. Default is 3.
group.file	a string of characters to indicate the file name assigning groups and colors to each supplementary element of the sup matrix.

Details

Metric multidimensional scaling is a statistical analysis technique aimed at analyzing a matrix of distances between 'active' elements. MDS maps the elements onto a low dimensional space (usually 2D or 3D). In this space, elements are represented by points whose respective distances best approximate the initial distance. In addition, after the metric MDS analysis of active elements, the `mmds.project` function allows projecting supplementary elements onto the active space in the context of the R environment. The active space is defined only by the MDS analysis of the active elements. The position of supplementary elements onto the active space depends only on their distances to active elements.

`sup` must have some characteristics:

- `sup` represents the matrix of pairwise distances between supplementary (rows) and active (columns) elements and does not have to be symmetric. The number of supplementary elements may be lower or higher than the number of active objects. The names of supplementary elements must be placed in the left column of `sup`.

The method for the computation of metric MDS projection of supplementary data is described by Abdi (2007). Briefly, if *N* is the number of active sequences and *D* is the *N* by *N* matrix of the squared distances computed from the active matrix, the `mmds` function performs the following steps:

(1) Transforms **D** into a cross-product matrix **S**:

$$\boldsymbol{S} = -0.5[\boldsymbol{I} - \boldsymbol{m}^T\mathbf{1}] \times \boldsymbol{D} \times [\boldsymbol{I} - \boldsymbol{m}^T\mathbf{1}]^T$$

where **I** is the *N* by *N* identity matrix, **m** is the *N* by *1* matrix mass, where each mass equal to $\frac{1}{N}$ and **1** is an *N* by *N* matrix of ones.

(2) Transforms ***S*** into a factor matrix ***F***:

$$\boldsymbol{F} = \boldsymbol{M}^{\frac{-1}{2}}\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$$

where **M** is the diag{**m**}, ***U*** is the eigenvector matrix and $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues, such as ***S*** = $\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\boldsymbol{T}}$, where $^{\boldsymbol{T}}$ denotes the transposition operation.

The eigenvectors of ***S***, also called principal components (whose number is smaller or equal to ***N***), form the active space. ***F*** gives the coordinates of the active elements in this space.

The supplementary elements are projected onto the active space as described below. If N_{sup} is the number of supplementary sequences and if D_{sup} is the N_{sup} by *N* matrix of the squared distances, the `mmds.project` function performs the following steps :

(3) Transforms $\boldsymbol{D_{sup}}$ into a cross-product matrix $\boldsymbol{S_{sup}}$:

$$\boldsymbol{S_{sup}} = -0.5[\boldsymbol{I} - \boldsymbol{m}^T\mathbf{1}] \times [\boldsymbol{D_{sup}}^{\boldsymbol{T}} - \boldsymbol{D}\boldsymbol{m}\mathbf{1}_{\boldsymbol{sup}}^{\boldsymbol{T}}],$$

where $\mathbf{1}_{\boldsymbol{sup}}$ is an $\boldsymbol{N_{sup}}$ by **N** matrix of ones.

(4) Transforms $\boldsymbol{S_{sup}}$ into a factor matrix $\boldsymbol{F_{sup}}$:

$$\boldsymbol{F_{sup}} = \boldsymbol{S_{sup}}^{\boldsymbol{T}}\boldsymbol{F}\boldsymbol{\Lambda}^{-1}.$$

$\boldsymbol{F_{sup}}$ gives the coordinates of the supplementary elements in the active space.

Value

A object of class 'project', which is a named list of three elements:

`coord`	a named numeric matrix representing the coordinates of active elements.
`group`	a named string matrix representing the differents groups of elements and the associate color (Default is 'NoGroup' and 'black').
`col`	a named string matrix representing, foreach named elements, the associate group and color (Default is 'NoGroup' and 'black).

Note

If `sup` do not contain names:
A tag "S" followed by an incremented number names the rows of `sup`.
The columns of `sup` are named as the rows of `mmds$D`.

Author(s)

Julien Pele and Jean-Michel Becu

References

Abdi H (2007) Metric multidimensional scaling. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics.* Thousand Oaks (CA): Sage. pp. 598-605.

For further reading on projection of supplementary elements:

Gower JC (1968) adding a point to vector diagrams in multivariate analysis. *Biometrika* **55**:582-585.

Trosset MW and Pribe CE (2008) The out-of-sample problem for classical multidimensional scaling. *Computational statistics & Data analysis* **52**:4635-4642.

Pele J, Abdi H, Moreau M, Thybert D and Chabbert M (2011) Multidimensional scaling reveals the main evolutionary pathways of class A G-protein-coupled receptors. *PloS ONE* **6**:e19094.

See Also

`cmdscale` function from `stats` package.
`dudi.pco`, `suprow` and `supcol` functions from `ade4` package.
`PCA` function from `FactoMineR` package.

Examples

```
# performing metric MDS of human GPCRs with projection of
# GPCRs from D. melanogaster as supplementary elements:
data(gpcr)
active <- gpcr$dif$sapiens.sapiens
sup <- gpcr$dif$melanogaster.sapiens
mmds_active<-mmds(active)
mmds_sup <- mmds.project(mmds_active,sup)
```

random.msa *Random Alignment*

Description

Builds a multiple sequence alignment (MSA) of random sequences.

Usage

```
random.msa(nb.seq = 100, id = "SEQ", nb.pos = 100, gap = FALSE,
aa.strict = FALSE, align = NULL, align.replace = TRUE)
```

Arguments

nb.seq	a numeric value indicating the number of sequences in the random MSA. Default is 100.
id	a string of characters used to tag each sequence name. Default is "SEQ". An incremented number is attached to this tag to name each sequence.
nb.pos	a numeric value indicating the length of each sequence in the random MSA. Default is 100.
gap	a logical value indicating whether the gap character should be considered as a supplementary symbol (TRUE) or not (FALSE). Default is FALSE.
aa.strict	a logical value indicating whether only strict amino acids should be taken into account (TRUE) or not (FALSE). Default is FALSE.
align	an object of class 'align', obtained from import.fasta or import.msf function. If this parameter is not NULL, the composition of the output sequences is based on the composition of the input sequences. Default is NULL.
align.replace	a logical value indicating random drawing with replacement (TRUE) or without replacement (FALSE) of characters present in align. Default is FALSE.

Details

random.msa may be used to compare a reference MSA to a random MSA. The random MSA must have the same characteristics as the reference MSA (same number of sequences of same length).

A mmds procedure can be applied to the random MSA to assess the amount of variance due to random mutations in the reference MSA.

Value

A named list whose objects correspond to random sequences.

Note

The subset function is used for random selection of the amino acids. If a truly random procedure is needed, see random package.

Author(s)

Julien Pele

References

For an application of random MSA see :

Pele J, Abdi H, Moreau M, Thybert D and Chabbert M (2011) Multidimensional scaling reveals the main evolutionary pathways of class A G-protein-coupled receptors. *PLoS ONE* **6**: e19094. doi:10.1371.

See Also

`permutation` and `synsequence` functions from `seqinr` package.

Examples

```
# generating a random sequence alignment with the same characterics
# as human GPCRs:
aln <- import.fasta(system.file("msa/human_gpcr.fa", package = "bios2mds"))
nb.seq <- length(aln)
nb.pos <- length(aln[[1]])
aln.random <- random.msa(nb.seq = nb.seq, nb.pos = nb.pos)
```

---

scree.plot *Plots the eingenvalues of an MDS analysis*

Description

Displays a bar plot of the eigenvalues obtained by MDS

Usage

```
scree.plot(x, lab = FALSE, title = NULL, xlim = NULL,
    ylim = NULL, new.plot = TRUE, pdf.file = NULL)
```

Arguments

`x`	a numeric vector representing the raw or the relative eigenvalues of each component.
`lab`	a boolean indicating whether bar labels should be displayed (TRUE) or not (FALSE). Default is FALSE.
`title`	a character string representing the title of the plot. Default is "Scree plot".
`xlim`	a numeric vector representing the range of the x values. Default is full range.
`ylim`	a numeric vector representing the range of the y values. Default is full range.
`new.plot`	a boolean indicating whether a new graphical device should be created (TRUE) or not (FALSE). Default is TRUE.
`pdf.file`	a string indicating the name and directory of the pdf graph outfile. Default is NULL. If this parameter is not NULL, the parameter `new.plot` is inactivated.

Details

A scree plot is a method for determining the optimal number of components useful to describe the data in the context of metric MultiDimensional Scaling (MDS). The scree plot is an histogram showing the eigenvalues of each component. The relative eigenvalues express the ratio of each eigenvalue to the sum of the eigenvalues. The relative eigenvalue of a component gives the proportion of the data variance explained by this component.

The aim is to evaluate the number of components required to capture most information contained in the data. In a scree plot, the relative eigenvalues decrease when the component number increases. The 'elbow' of the plot determines the optimal number of components to describe the data (usually the components before the 'elbow').

Value

Produces a bar plot on the active graphical device.

Note

The scree plot is not an exclusive method to determine the optimal number of components. A shepard plot, which is a scatterplot of the scaled MDS distances against the original distance data, can be another solution. See `shepard` function from `MASS` package.

Author(s)

Julien Pele

See Also

`plot.pca.scree` function from `bio3d` package.
`goodness.metaMDS` function from `vegan` package.

Examples

```
# displaying the scree plot of the MDS analysis of human GPCRs
data(gpcr)
active <- gpcr$dif$sapiens.sapiens
mmds1 <- mmds(active, pc = 5)
scree.plot(mmds1$eigen.perc, lab = TRUE, title = "Scree plot of metric MDS")
```

---

sil.score *Silhouette score*

Description

Computes silhouette scores for multiple runs of K-means clustering.

Usage

```
sil.score(mat, nb.clus = c(2:13), nb.run = 100, iter.max = 1000,
method = "euclidean")
```

Arguments

mat	a numeric matrix representing the coordinates of the elements.
nb.clus	a numeric vector indicating the range of the numbers of clusters. Default is c(2:13).
nb.run	a numeric value indicating the number of runs. Default is 100.
iter.max	a numeric value indicating the maximum number of iterations for K-means clustering. Default is 1000.
method	a string of characters to determine the distance measure. This should be one of "euclidean" , "maximum", "manhattan", "canberra" or "binary". Default is "euclidean".

Details

Silhouettes are a general graphical aid for interpretation and validation of cluster analysis. This technique is available through the `silhouette` function (`cluster` package). In order to calculate silhouettes, two types of data are needed:

- the collection of all distances between objects. These distances are obtained from application of `dist` function on the coordinates of the elements in `mat` with argument `method`.
- the partition obtained by the application of a clustering technique. In `sil.score` context, the partition is obtained from the `Kmeans` function (`amap` package) with argument `method` which indicates the cluster to which each element is assigned.

For each element, a silhouette value is calculated and evaluates the degree of confidence in the assignment of the element:

- well-clustered elements have a score near 1,
- poorly-clustered elements have a score near -1.

Thus, silhouettes indicates the objects that are well or poorly clustered. To summarize the results, for each cluster, the silhouettes values can be displayed as an **average silhouette width**, which is the mean of silhouettes for all the elements assigned to this cluster. Finally, the **overall average silhouette** width is the mean of average silhouette widths of the different clusters.

Silhouette values offer the advantage that they depend only on the partition of the elements. As a consequence, silhouettes can be used to compare the output of the same clustering algorithm applied to the same data but for different numbers of clusters. A range of numbers of clusters can be tested, with the `nb.clus` argument. The optimal number of clusters is reached for the maximum of the overall silhouette width. This means that the clustering algorithm reaches a strong clustering structure. However, for a given number of clusters, the cluster assignment obtained by different K-means runs can be different because the K-means procedure assigns random initial centroids for each run. It may be necessary to run the K-means procedure several times, with the nb.run argument, to evaluate the uncertainty of the results. In that case, for each number of clusters, the mean of the overall average silhouettes for `nb.run` K-means runs is calculated. The maximum of this core gives the optimal number of clusters.

Value

A named numeric vector representing the silhouette scores for each number of clusters.

Note

`sil.score` requires `Kmeans` and `silhouette` functions from `amap` and `cluster` packages, respectively.

Author(s)

Julien Pele

References

Rousseeuw PJ (1987) Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, **20**:53-65.

Lovmar L, Ahlford A, Jonsson M and Syvanen AC (2005) Silhouette scores for assessment of SNP genotype clusters. *BMC Genomics*, **6**:35.

Guy B, Vasyl P, Susmita D and Somnath D (2008) clValid: An R Package for Cluster Validation. *Journal of Statistical Software*, **25**.

See Also

`connectivity` and `dunn` functions from `clValid` package.
`silhouette` function from `cluster` package.

Examples

```
# calculating silhouette scores for K-means clustering of human GPCRs:
data(gpcr)
active <- gpcr$dif$sapiens.sapiens
mds <- mmds(active)
sil.score1 <- sil.score(mds$coord, nb.clus = c(2:10),
 nb.run = 100, iter.max = 100)
barplot(sil.score1)
```

---

sub.mat *Amino acid substitution matrices*

Description

Contains eight amino acid substitution matrices, imported from version 9.1 of the aaindex2 database and PAM matrix calculator of Wageningen Bioinformatics Webportal.

Usage

```
data(sub.mat)
```

Format

A named list with eight elements corresponding to a 20 by 20 named matrix. Rows and columns names correspond to the twenty strict amino acids.

Details

PAM40 matrix was produced by "pam" Version 1.0.7

PAM120 matrix was produced by "pam" Version 1.0.7

PAM160 matrix was produced by "pam" Version 1.0.7

PAM250 log odds matrix for 250 PAMs (Dayhoff et al., 1978)

BLOSUM30 substitution matrix (Henikoff-Henikoff, 1992)

BLOSUM45 substitution matrix (Henikoff-Henikoff, 1992)

BLOSUM62 substitution matrix (Henikoff-Henikoff, 1992)

BLOSUM80 substitution matrix (Henikoff-Henikoff, 1992)

GONNET substitution matrix (GONNET et al., 1992)

JTT substitution matrix (Jones et al., 1992)

JTT_TM transmembrane protein exchange matrix (Jones et al., 1994)

PHAT substitution matrix built from hydrophobic and transmembrane regions of the Blocks database (Ng et al., 2000)

Source

The matrices were downloaded from the AAindex database at https://www.genome.jp/aaindex/ or were calculated on the PAM server at http://www.bioinformatics.nl/tools/pam.html.

References

Kawashima S and Kanehisa M (2000) AAindex: amino acid index database. *Nucleic Acids Res* **28**:374.

Dayhoff MO, Schwartz R and Orcutt BC (1978) A model of Evolutionary Change in Proteins. Atlas of protein sequence and structure (volume 5, supplement 3 ed.). Nat. Biomed. Res. Found.. pp. 345-358.

Henikoff S and Henikoff JG (1992) Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A* **89**:10915-9.

Jones DT, Taylor WR and Thornton JM (1992) The rapid generation of mutation data matrices from protein sequences. *Comput Appl. Biosci* **8**:275-282.

Jones DT, Taylor WR and Thornton JM (1994) A mutation data matrix for transmembrane proteins. *FEBS Lett* **339**:269-75.

Ng PC, Henikoff JG and Henikoff S (2000) PHAT: a transmembrane-specific substitution matrix. Predicted hydrophobic and transmembrane. *Bioinformatics* **16**:760-6.

Gonnet GH, Cohen MA and Benner SA (1992) Exhaustive matching of the entire protein sequence database. *Science* **256**:1443-1445.

Examples

```
# loading
data(sub.mat)

# displaying PAM40:
sub.mat$PAM40
```

`write.mmds.pdb` *Writes PDB file*

Description

Writes MDS coordinates in the Protein Data Bank format for visualization with a molecular graphics viewer.

Usage

```
write.mmds.pdb(x,project = NULL, axis = c(1, 2, 3), file.pdb = "R.pdb", file.pml=NULL)
```

Arguments

`x`	an object of class 'mmds', obtained from `mmds` function.
`project`	an object of class 'project', obtained from `mmds.project` function, or a list of object of class 'project'.
`axis`	a numeric vector of length three the principal components to be displayed. Default is c(1, 2, 3).
`file.pdb`	a string of characters indicating the output PDB file name. Default is "R.pdb".
`file.pml`	a string of characters indicating the output pml file name for visualization with Pymol. If this parameter is not NULL, the pml file will be written. Default is NULL.

Details

The elements can be visualized in three dimensions (3D) with a molecular viewer as Pymol or Rasmol. If `x` contains active and supplementary elements, the active and supplementary elements are numbered from 1 and from 5001, respectively. If group is not NULL, the assignment of an element to a group is indicated by the chain name from *A* for the first group to *Z* when the maximum number of groups, 26, is reached.

The pml file allows a fancy visualization of the PDB file with the Pymol molecular viewer. The user must first open the PDB file with Pymol, then run the pml file. The active and inactive elements will be displayed as spheres and crosses, respectively, with coloring based on the user-provided colors with the `col.group` function.

Value

Produces a PDB file from the MDS coordinates, with the elements numbered in the order of the MSA file and the groups corresponding to the chain numbers. Optionnaly, produces a pml file to add color and group selection in pymol with the pdb file.

Author(s)

Julien Pele and Jean-Michel Becu

References

https://www.rcsb.org

See Also

write.pdb function from bio3d package.

Examples

```
# writing the first three MDS coordinates of human GPCRs in a PDB file
wd <- tempdir()
#wd <- getwd()
file1 <- file.path(wd,"sapiens.pdb")
file2 <- file.path(wd,"sapiens.pml")
data(gpcr)
write.mmds.pdb(gpcr$mmds$sapiens.active,file.pdb=file1,file.pml=file2)
```

Index

∗ **clustering**
  kmeans.run, 16
  sil.score, 37
∗ **datasets**
  gpcr, 11
  sub.mat, 39
∗ **distance**
  dif, 6
  dis, 7
  mat.dif, 17
  mat.dis, 18
∗ **exploratory analysis**
  mmds, 20
  mmds.project, 32
∗ **extraction**
  extract.cluster, 10
∗ **mmds**
  col.group, 5
∗ **package**
  bios2mds-package, 2
∗ **plot**
  mmds.2D.multi, 22
  mmds.2D.plot, 25
  mmds.plot, 31
  scree.plot, 36
∗ **utilities**
  random.msa, 35
∗ **write**
  write.mmds.pdb, 41

bios2mds *(*bios2mds-package*)*, 2
bios2mds-package, 2

col.group, *3*, 5

dif, 6
dis, 7

export.fasta, 9
extract.cluster, *3*, *9*, 10

gpcr, 11

import.fasta, *2*, *9, 10*, 13, *35*
import.msf, *2*, *9, 10*, 15, *35*

kmeans.run, *3*, *10*, 16

mat.dif, *2*, *11*, 17
mat.dis, *2*, *12*, 18
mmds, *2*, *5*, *12*, 20, *22*, *26*, *29*, *31, 32*, *35*, *41*
mmds.2D.multi, *3*, 22
mmds.2D.plot, *3*, 25, *31*
mmds.3D.plot, *3*, 29
mmds.plot, *3*, 31
mmds.project, *2*, *5*, *22*, *26*, *29*, *31*, 32, *41*

random.msa, 35

scree.plot, *3*, *31*, 36
sil.score, *3*, 37
sub.mat, *7*, *12*, *19*, 39

write.mmds.pdb, *3*, 41