Mfuzz
November 11, 2009

R topics documented:

acore

csSelection

fill.NA

filter.NA

filter.std

kmeans2.plot

kmeans2

mfuzzColorBar

Mfuzzgui

mfuzz.plot2

mfuzz.plot

mfuzz

overlap.plot

overlap

partcoeff

randomise

standardise2

standardise

top.count

yeast

yeast.table2

yeast.table

Index

acore

Extraction of alpha cores for soft clusters

Description

This function extracts genes forming the alpha cores of soft clusters

Usage

acore(eset, cl, min.acore=0.5)
cselection

Arguments

eset object of the class ExpressionSet.
cl An object of class flcust as produced by mfuzz.
min.acore minimum membership values of gene belonging to the cluster core.

Value

The function produces an list of alpha cores including genes and their membership values for the corresponding cluster.

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)

Examples

if (interactive()){
  ### Data loading and pre-processing
  data(yeast) # data set includes 17 measurements
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF)
  yeastF <- standardise(yeastF)

  ### Soft clustering and visualisation
  cl <- mfuzz(yeastF,c=20,m=1.25)
  acore.list <- acore(yeastF,cl=cl,min.acore=0.7)
}

cselection Repeated soft clustering for detection of empty clusters

Description

This function performs repeated soft clustering for a range of cluster numbers c and reports the number of empty clusters detected.

Usage

cselection(eset,m,crange=seq(4,32,4),repeats=5,visu=TRUE,...)

Arguments

eset object of class ExpressionSet.
m value of fuzzy c-means parameter m.
crange range of number of clusters c.
repeats number of repeated clusterings.
visu If visu=TRUE plot of number of empty clusters is produced.
... additional arguments for underlying mfuzz.
Details

A soft cluster is considered as empty, if none of the genes has a corresponding membership value larger than 0.5.

Value

A matrix with the number of empty clusters detected is generated.

Note

The `cselection` function may help to determine an accurate cluster number. However, it should be used with care, as the determination remains difficult especially for short time series and overlapping clusters. A better way is likely to perform clustering with a range of cluster numbers and subsequently assess their biological relevance e.g. by GO analyses.

Author(s)

Matthias E. Futschik (http://www.cbme.ualg.pt/mfutschik_cbme.html)

References


L. Kumar and M. Futschik, Mfuzz: a software package for soft clustering of microarray data, Bioinformation, 2(1) 5-7, 2007

Examples

```r
if (interactive()){
  data(yeast)
  # Data pre-processing
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF)
  yeastF <- standardise(yeastF)
  #### parameter selection
  # Empty clusters should not appear
  cl <- mfuzz(yeastF,c=20,m=1.25)
  mfuzz.plot(yeastF,cl=cl,mfrow=c(4,5))
  # Note: The following calculation might take some time
  tmp <- cselection(yeastF,m=1.25,crange=seq(5,40,5),repeats=5,visu=TRUE)
  # derivation of number of non-empty clusters (crosses) from diagonal
  # line indicate appearance of empty clusters
  # Empty clusters might appear
  cl <- mfuzz(yeastF,c=40,m=1.25)
  mfuzz.plot(yeastF,cl=cl,mfrow=c(4,5))
}
```
fill.NA  Replacement of missing values

Description

Methods for replacement of replacing missing values. Missing values should be indicated by NA in the expression matrix.

Usage

fill.NA(eset, mode="mean", k=10)

Arguments

eset  object of the class ExpressionSet.
mode  method for replacement of missing values:
      • mean- missing values will be replaced by the mean expression value of the gene,
      • median- missing values will be replaced by the median expression value of the gene,
      • knn- missing values will be replaced by the averaging over the corresponding expression values of the k-nearest neighbours,
      • knnw- same replacement method as knn, but the expression values averaged are weighted by the distance to the corresponding neighbour

k  Number of neighbours, if one of the knn method for replacement is chosen (knn, knnw).

Value

The function produces an object of the ExpressionSet class with missing values replaced.

Note

The replacement methods knn and knnw can computationally intensive for large gene expression data sets. It may be a good idea to run these methods as a ‘lunchtime’ or ‘overnight’ job.

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik) and Lokesh Kumar

Examples

if (interactive()):
data(yeast)  # data set includes 17 measurements
yeastF <- filter.NA(yeast)
yeastF <- fill.NA(yeastF)
}
**filter.NA**

Filtering of genes based on number of non-available expression values.

**Description**

This function can be used to exclude genes with a large number of expression values not available.

**Usage**

```r
filter.NA(eset, thres=0.25)
```

**Arguments**

- `eset`: object of the class “ExpressionSet”.
- `thres`: threshold for excluding genes. If the percentage of missing values (indicated by NA in the expression matrix) is larger than `thres`, the corresponding gene will be excluded.

**Value**

The function produces an object of the ExpressionSet class. It is the same as the input `eset` object, except for the genes excluded.

**Author(s)**

Matthias E. Futschik ([http://itb.biologie.hu-berlin.de/~futschik](http://itb.biologie.hu-berlin.de/~futschik))

**Examples**

```r
if (interactive()){
  data(yeast) # data set includes 17 measurements
  yeastF <- filter.NA(yeast) # genes are excluded if more than 4 measurements are missing
}
```

---

**filter.std**

Filtering of genes based on their standard deviation.

**Description**

This function can be used to exclude genes with low standard deviation.

**Usage**

```r
filter.std(eset, min.std, visu=TRUE)
```
Arguments

- **eset**: object of the class `ExpressionSet`.
- **min.std**: threshold for minimum standard deviation. If the standard deviation of a gene’s expression is smaller than `min.std` the corresponding gene will be excluded.
- **visu**: If `visu` is set to `TRUE`, the ordered standard deviations of genes’ expression values will be plotted.

Value

The function produces an object of the `ExpressionSet` class. It is the same as the input `eset` object, except for the genes excluded.

Note

As soft clustering is noise robust, pre-filtering can usually be avoided. However, if the number of genes with small expression changes is large, such pre-filtering may be necessary to reduce noise.

Author(s)

Matthias E. Futschik ([http://itb.biologie.hu-berlin.de/~futschik](http://itb.biologie.hu-berlin.de/~futschik))

Examples

```r
data(yeast) # data set includes 17 measurements
yeastF <- filter.NA(yeast) # filtering of genes based on missing values
yeastF <- filter.std(yeastF,min.std=0.3) # filtering of genes based on standard deviation
```

Description

This function visualises the clusters produced by `kmeans2`.

Usage

```r
kmeans2.plot(eset,kl,mfrow=c(1,1))
```

Arguments

- **eset**: object of the class "ExpressionSet".
- **kl**: list produced by `kmeans2`.
- **mfrow**: determines splitting of graphic window.

Value

The function displays the temporal profiles of clusters detected by k-means.

Author(s)

Matthias E. Futschik ([http://itb.biologie.hu-berlin.de/~futschik](http://itb.biologie.hu-berlin.de/~futschik))
### kmeans2

**Description**

This function is a wrapper function for `kmeans` of the `e1071` package. It performs hard clustering of genes based on their expression values using the k-means algorithm.

**Usage**

```r
kmeans2(eset, k, iter.max=100)
```

**Arguments**

- `eset` : object of the class `ExpressionSet`.
- `k` : number of clusters.
- `iter.max` : maximal number of iterations.

**Value**

An list of clustering components (see `kmeans`).

**Author(s)**

Matthias E. Futschik ([http://itb.biologie.hu-berlin.de/~futschik](http://itb.biologie.hu-berlin.de/~futschik))

**See Also**

- `kmeans`

**Examples**

```r
if (interactive()){
  data(yeast)
  # Data pre-processing
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF)
  yeastF <- standardise(yeastF)

  # K-means clustering and visualisation
  kl <- kmeans2(yeastF,k=20)
  kmeans2.plot(yeastF,kl=kl,mfrow=c(2,2))
}
```
# K-means clustering and visualisation
kl <- kmeans2(yeastF, k=20)
kmeans2.plot(yeastF, kl=kl, mfrow=c(2, 2))

--

### mfuzzColorBar

**Plots a colour bar**

**Description**

This function produces a (separate) colour bar for graphs produced by mfuzz.plot

**Usage**

```r
mfuzzColorBar(col, horizontal=FALSE, ...)
```

**Arguments**

- `col` vector of colours used. If missing, the same vector as the default vector for mfuzz.plot is used. If col="fancy", an alternative color palette is used (see mfuzz.plot2).
- `horizontal` If TRUE, a horizontal colour bar is generated, otherwise a vertical one will be produced.
- `...` additional parameter passed to maColorBar (see also example in mfuzz.plot2)

**Author(s)**

Matthias E. Futschik [http://itb.biologie.hu-berlin.de/~futschik](http://itb.biologie.hu-berlin.de/~futschik)

**See Also**

- maColorBar

**Examples**

```r
if (interactive()){
  X11(w=1.5, h=5);
  par(mar=c(1, 1, 1, 5))
  mfuzzColorBar()
  mfuzzColorBar(col="fancy", main="Membership value")
}
```
Description

The function Mfuzzgui provides a graphical user interface for clustering of microarray data and visualisation of results. It is based on the functions of the Mfuzz package.

Usage

Mfuzzgui()

Details

The function Mfuzzgui launches a graphical user interface for the Mfuzz package. It is based on Tk widgets using the R TclTk interface by Peter Dalgaard. It also employs some pre-made widgets from the tkWidgets Bioconductor-package by Jianhua Zhang for the selection of objects/files to be loaded.

Mfuzzgui provides a convenient interface to most functions of the Mfuzz package without restriction of flexibility. An exception is the batch processes such as partcoeff and cselection routines which are used for parameter selection in fuzzy c-means clustering of microarray data. These routines are not included in Mfuzzgui. To select various parameters, the underlying Mfuzz routines may be applied.

Usage of Mfuzzgui does not require assumes an pre-built exprSet object but can be used with tab-delimited text files containing the gene expression data. Note, however, that the clustering is based on the the ordering of samples (arrays) as of the columns in the expression matrix of the exprSet object or in the uploaded table, respectively. Also, replicated arrays in the expression matrix (or table) are treated as independent by the mfuzz function and, thus, should be averagered prior to clustering.

For a overview of the functionality of Mfuzzgui, please refer to the package vignette. For a description of the underlying functions, please refer to the Mfuzz package.

Value

Mfuzzgui returns a tclObj object.

Note

The newest versions of Mfuzzgui can be found at the Mfuzz webpage (http://itb.biologie.hu-berlin.de/~futschik/software/R/Mfuzz).

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik) and Lokesh Kumar
References


See Also

mfuzz

mfuzz.plot2
Plotting results for soft clustering with additional options

Description

This function visualises the clusters produced by mfuzz. It is similar to mfuzz.plot, but offers more options for adjusting the plots.

Usage

mfuzz.plot2(eset, cl, mfrow=c(1,1), colo=min.mem=0, time.labels, x11=TRUE, ax.col="black", bg = "white", col.axis="black", col.lab="black", col.main="black", col.sub="black", cex.main=2, Xwidth=5, Xheight=5, single=FALSE,...)

Arguments

eset object of the class ExpressionSet.
cl object of class flclust.
mfrow determines splitting of graphic window. Use mfrow=NA if layout is used (see example).
colo color palette to be used for plotting. If the color argument remains empty, the default palette is used. If the colo = "fancy", an alternative (fancier) palette will be used.
min.mem Genes with membership values below min.mem will not be displayed.
time.labels labels can be given for the time axis.
x11 If TRUE, a new window will be open for plotting.
ax.col Color of axis line.
bg Background color.
col.axis Color for axis annotation.
col.lab Color for axis labels.
col.main Color for main titles.
col.sub Color for sub-titles.
mfuzz.plot

Plotting results for soft clustering

Description

This function visualises the clusters produced by mfuzz.

Usage

mfuzz.plot(eset, cl, mfrow=c(1,1), colo, min.mem=0, time.labels, new.window=TRUE)
mfuzz

Function for soft clustering based on fuzzy c-means.

Description

This function is a wrapper function for cmeans of the e1071 package. It performs soft clustering of genes based on their expression values using the fuzzy c-means algorithm.

Usage

mfuzz(eset, centers, m, ...)

Arguments

eset object of the class ExpressionSet.
cl object of class flclust.
mfrow determines splitting of graphic window.
colo color palette to be used for plotting. If the color argument remains empty, the default palette is used.
min.mem Genes with membership values below min.mem will not be displayed.
time.labels labels can be given for the time axis.
new.window should a new window be opened for graphics.

Value

The function generates plots where the membership of genes is color-encoded.

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)

Examples

if (interactive()){
  data(yeast)
  # Data pre-processing
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF)
  yeastF <- standardise(yeastF)
  # Soft clustering and visualisation
  cl <- mfuzz(yeastF,c=20,m=1.25)
  mfuzz.plot(yeastF,cl=cl,mfrow=c(2,2))
  # display of cluster cores with alpha = 0.5
  mfuzz.plot(yeastF,cl=cl,mfrow=c(2,2),min.mem=0.5)
  # display of cluster cores with alpha = 0.7
  mfuzz.plot(yeastF,cl=cl,mfrow=c(2,2),min.mem=0.7)
}

mfuzz
**Arguments**

- **eset**: object of the class “ExpressionSet”.
- **centers**: number of clusters.
- **m**: fuzzification parameter.
- **...**: additional parameters for `cmeans`.

**Details**

This function is the core function for soft clustering. It groups genes based on the Euclidean distance and the c-means objective function which is a weighted square error function. Each gene is assigned a membership value between 0 and 1 for each cluster. Hence, genes can be assigned to different clusters in a gradual manner. This contrasts hard clustering where each gene can belongs to a single cluster.

**Value**

An object of class `flcust` (see `cmeans`) which is a list with components:

- **centers**: the final cluster centers.
- **size**: the number of data points in each cluster of the closest hard clustering.
- **cluster**: a vector of integers containing the indices of the clusters where the data points are assigned to for the closest hard clustering, as obtained by assigning points to the (first) class with maximal membership.
- **iter**: the number of iterations performed.
- **membership**: a matrix with the membership values of the data points to the clusters.
- **withinererror**: the value of the objective function.
- **call**: the call used to create the object.

**Note**

Note that the clustering is based solely on the `exprs` matrix and no information is used from the `phenoData`. In particular, the ordering of samples (arrays) is the same as the ordering of the columns in the `exprs` matrix. Also, replicated arrays in the `exprs` matrix are treated as independent by the `mfuzz` function i.e. they should be averaged prior to clustering or placed into different distinct “ExpressionSet” objects.

**Author(s)**

Matthias E. Futschik (http://www.cbme.ualg.pt/mfutschik_cbme.html)

**References**


L. Kumar and M. Futschik, Mfuzz: a software package for soft clustering of microarray data, Bioinformation, 2(1) 5-7, 2007

**See Also**

`cmeans`
Examples

```r
if (interactive()){
  data(yeast)
  # Data pre-processing
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF) # for illustration only; rather use knn method
  yeastF <- standardise(yeastF)

  # Soft clustering and visualisation
  cl <- mfuzz(yeastF,c=20,m=1.25)
  mfuzz.plot(yeastF,cl=cl,mfrow=c(2,2))

  # Plotting center of cluster 1
  X11(); plot(cl[[1]][1,],type="l",ylab="Expression")

  # Getting the membership values for the first 10 genes in cluster 1
  cl[[4]][1:10,1]
}
```

overlap.plot `Visualisation of cluster overlap and global clustering structure`

Description

This function visualises the cluster overlap produced by `overlap`.

Usage

```r
overlap.plot(cl, overlap, thres=0.1, scale=TRUE, magni=30, P=NULL)
```

Arguments

- `cl`: object of class “flclust"
- `overlap`: matrix of cluster overlap produced by `overlap`
- `thres`: threshold for visualisation. Cluster overlaps below the threshold will not be visualised.
- `scale`: Scale parameter for principal component analysis by `prcomp`
- `magni`: Factor for increase the line width for cluster overlap.
- `P`: Projection matrix produced by principal component analysis.

Value

A plot is generated based on a principal component analysis of the cluster centers. The overlap is visualised by lines with variable width indicating the strength of the overlap. Additionally, the matrix of principal components is returned. This matrix can be re-used for other projections to compare the overlap and global cluster structure of different clusterings.

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)
See Also

\texttt{prcomp}

Examples

\begin{verbatim}
if (interactive()){
  data(yeast)
  # Data pre-processing
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF)
  yeastF <- standardise(yeastF)

  # Soft clustering
  cl <- mfuzz(yeastF,c=20,m=1.25)
  X11();mfuzz.plot(yeastF,cl=cl,mfrow=c(4,5))
  O <- overlap(cl)
  X11();Ptmp <- overlap.plot(cl,over=O,thres=0.05)

  # Alternative clustering
  cl <- mfuzz(yeastF,c=10,m=1.25)
  X11();mfuzz.plot(yeastF,cl=cl,mfrow=c(3,4))
  O <- overlap(cl)
  X11();overlap.plot(cl,over=O,P=Ptmp,thres=0.05)
  # visualisation based on principal compents from previous projection
}
\end{verbatim}

\texttt{overlap} \hspace{2cm} \textit{Calculation of the overlap of soft clusters}

Description

This function calculates the overlap of clusters produced by \texttt{mfuzz}.

Usage

\texttt{overlap(cl)}

Arguments

\begin{description}
  \item[cl] object of class \texttt{flclust}
\end{description}

Value

The function generates a matrix of the normalised overlap of soft clusters. The overlap indicates the extent of "shared" genes between clusters. For a mathematical definition of the overlap, see the vignette of the package or the reference below.

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)
partcoef

Calculation of the partition coefficient matrix for soft clustering

Description

This function calculates partition coefficient for clusters within a range of cluster parameters. It can be used to determine the parameters which lead to uniform clustering.

Usage

partcoef(eset, crange=seq(4,32,4), mrange=seq(1.05,2,0.1), ...)

Arguments

eset : object of class “ExpressionSet”.

creange : range of number of clusters c.
mrange : range of clustering parameter m.

... : additional arguments for underlying mfuzz.

Details

Introduced by Bezdek (1981), the partition coefficient F is defined as the sum of squares of values of the partition matrix divided by the number of values. It is maximal if the partition is hard and reaches a minimum for U=1/c when every gene is equally assigned to every cluster.

It is well-known that the partition coefficient tends to decrease monotonically with increasing n. To reduce this tendency we defined a normalized partition coefficient where the partition for uniform partitions are subtracted from the actual partition coefficients (Futschik and Kasabov, 2002).
**randomise**

**Value**

The function generates the matrix of partition coefficients for a range of \( c \) and \( m \) values. It also produces a matrix of normalised partition coefficients as well as a matrix with partition coefficient for uniform partitions.

**Author(s)**

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)

**References**

1. J.C.Bezdek, Pattern recognition with fuzzy objective function algorithms, Plenum, 1981

**Examples**

```r
if (interactive()){
  data(yeast)
  # Data pre-processing
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF)
  yeastF <- standardise(yeastF)
  
  ### parameter selection
  yeastFR <- randomise(yeastF)
  cl <- mfuzz(yeastFR,c=20,m=1.1)
  mfuzz.plot(yeastFR,cl=cl,mfrow=c(4,5)) # shows cluster structures (non-uniform partition)

  tmp <- partcoef(yeastFR) # This might take some time.
  F <- tmp[[1]];F.n <- tmp[[2]];F.min <- tmp[[3]]
  
  # Which clustering parameters result in a uniform partition?
  F > 1.01 * F.min
  cl <- mfuzz(yeastFR,c=20,m=1.25) # produces uniform partion
  mfuzz.plot(yeastFR,cl=cl,mfrow=c(4,5)) # uniform coloring of temporal profiles indicates uniform partition
}
```

---

**randomise**

**Randomisation of data**

**Description**

This function randomise the time order for each gene separately.

**Usage**

`randomise(eset)`
standardise2

Arguments

eset object of the class ExpressionSet.

Value

The function produces an object of the ExpressionSet class with randomised expression data.

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)

Examples

data(yeast) # data set includes 17 measurements
yeastR <- randomise(yeast)

standardise2 Standardization in regards to selected time-point

Description

Standardisation of the expression values of every gene is performed, so that the expression values at a chosen time point are zero and the standard deviations are one.

Usage

standardise2(eset,timepoint=1)

Arguments

eset object of the class ExpressionSet.

timepoint integer: which time point should have expression values of zero.

Value

The function produces an object of the ExpressionSet class with standardised expression values.

Author(s)

Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)

Examples

if (interactive()){  
data(yeast)  
# Data pre-processing  
yeastF <- filter.NA(yeast)  
yeastF <- fill.NA(yeastF)  
yeastF <- standardise2(yeastF,timepoint=1)

# Soft clustering and visualisation  
cl <- mfuzz(yeastF,c=20,m=1.25)  
mfuzz.plot(yeastF,cl=cl,mfrow=c(4,5))  
}
**standardise**

Standardization of microarray data for clustering.

**Description**

Standardisation of the expression values of every gene is performed, so that the average expression value for each gene is zero and the standard deviation is one.

**Usage**

```r
standardise(eset)
```

**Arguments**

- `eset`: object of the class `ExpressionSet`.

**Value**

The function produces an object of the ExpressionSet class with standardised expression values.

**Author(s)**

Matthias E. Futschik ([http://itb.biologie.hu-berlin.de/~futschik](http://itb.biologie.hu-berlin.de/~futschik))

**Examples**

```r
if (interactive()){  
  data(yeast)  
  # Data pre-processing  
  yeastF <- filter.NA(yeast)  
  yeastF <- fill.NA(yeastF)  
  yeastF <- standardise(yeastF)  

  # Soft clustering and visualisation  
  cl <- mfuzz(yeastF,c=20,m=1.25)  
  mfuzz.plot(yeastF,cl=cl,mfrow=c(4,5))
}
```

---

**top.count**

Determines the number for which each gene has highest membership value in all cluster

**Description**

This function calculates the number for which each gene appears to have the top membership score in the partition matrix of clusters produced by `mfuzz`.

**Usage**

```r
top.count(cl)
```
Arguments

cl object of class “flclust”

Value

The function generates a vector containing a count for each gene, which is just the number of times that particular gene has acquired the top membership score.

Author(s)

Lokesh Kumar and Matthias E. Futschik (http://itb.biologie.hu-berlin.de/~futschik)

Examples

if (interactive()){
  data(yeast)
  # Data pre-processing
  yeastF <- filter.NA(yeast)
  yeastF <- fill.NA(yeastF)
  yeastF <- standardise(yeastF)

  # Soft clustering and visualisation
  cl <- mfuzz(yeastF,c=20,m=1.25)
  top.count(cl)
}

yeast

Gene expression data of the yeast cell cycle

Description

The data contains gene expression measurements for 3000 randomly chosen genes of the yeast mutant cdc28 as performed and described by Cho et al. For details, see the reference.

Usage

data(yeast)

Format

An object of class “ExpressionSet”.

Source

The data was downloaded from Yeast Cell Cycle Analysis Project webside and converted to an ExpressionSet object.

References

yeast.table2  Gene expression data of the yeast cell cycle as table

Description

The data serves as an example for the format required to upload tables with expression data into Mfuzzgui. The first row contains the names of the samples and the first column contains unique identifiers for genes. To input measurement time and gene names, refer to yeast.table.

The exemplary tables can be found in the data sub-folder of the Mfuzzgui package.

References


See Also

yeast.table

eyeast.table  Gene expression data of the yeast cell cycle as table

Description

The data serves as an example for the format required for uploading tables with expression data into Mfuzzgui. The first row contains the names of the samples, the second row contains the measured time points. Note that “TIME” has to placed in the first field of the second row.

The first column contains unique identifiers for genes; optionally the second row can contain gene names if “GENE.NAMES” is in the second field in the first row.

An example for an table without optional fields is the dataset yeast.table2.

The exemplary tables can be found in the data sub-folder of the Mfuzzgui package.

References


See Also

yeast.table2
Index

*Topic cluster
  cselection, 2
  kmeans2, 6
  mfuzz, 12
  Mfuzzgui, 8
  overlap, 14
  partcoeff, 15
  top.count, 19

*Topic datasets
  yeast, 20
  yeast.table, 21
  yeast.table2, 20

*Topic hplot
  kmeans2.plot, 6
  mfuzz.plot, 11
  mfuzz.plot2, 9
  overlap.plot, 13

*Topic misc
  Mfuzzgui, 8

*Topic ts
  Mfuzzgui, 8

*Topic utilities
  acore, 1
  fill.NA, 3
  filter.NA, 4
  filter.std, 5
  mfuzzColorBar, 7
  randomise, 17
  standardise, 18
  standardise2, 17

  acore, 1
  cmeans, 12, 13
  cselection, 2
  fill.NA, 3
  filter.NA, 4
  filter.std, 5
  kmeans, 6, 7
  kmeans2, 6
  kmeans2.plot, 6
  maColorBar, 8

mfuzz, 9, 12
mfuzz.plot, 11
mfuzz.plot2, 9
mfuzzColorBar, 7
Mfuzzgui, 8
overlap, 14
overlap.plot, 13
partcoeff, 15
prcomp, 14
randomise, 17
standardise, 18
standardise2, 17
top.count, 19
yeast, 20
yeast.table, 20, 21
yeast.table2, 20, 21