Introduction to Mfuzz package and its graphical user interface

Matthias E. Futschik
SysBioLab, Universidade do Algarve
URL: *http://mfuzz.sysbiolab.eu*

and

Lokesh Kumar
Institute for Advanced Biosciences, Keio-University, Japan

May 1, 2024

Contents

1 Overview

Clustering is an important tool in gene expression data analysis - both on transcript as well as on protein level. This unsupervised classificatiion technique is commonly used to reveal structures hidden in large gene expression data sets. The vast majority of clustering algorithms applied so far produce *hard partitions* of the data, i.e. each gene or protein is assigned exactly to one cluster. *Hard* clustering is favourable if clusters are well separated. However, this is generally not the case for gene expression data, where gene/protein clusters frequently overlap. Additionally, hard clustering algorithms are often highly sensitive to noise.

To overcome the limitations of hard clustering, *soft* clustering can be applied offering several advantages to researchers [1, 2]. First, it generates accessible internal cluster structures, i.e. it indicates how well corresponding clusters represent genes/proteins. This additional information can be used for a refined search for regulatory elements. Second, the overall relation between clusters, and thus a global clustering structure, can be defined. Additionally, soft clustering is more noise robust and *a priori* pre-filtering of genes can be avoided. This prevents the exclusion of biologically relevant genes/proteins from the data analysis.

This vignette gives a short introduction to soft clustering using the *Mfuzz* package. It misses some features (such cluster stability) due to the size restrictions for Bioconductor vignettes. Additional, the graphical user interace (*Mfuzzgui* is introduced here. Note that we use microarray data to illustrate the functions of *Mfuzz*, but other data such as RNA-Seq or proteomics data can be analysed by *Mfuzz* as well. The data pre-processing, however, might need to be adjusted for other types of data. More information regarding this and other issues can be found in the *Questions and Answers* section on the *Mfuzz* webpage:
http://mfuzz.sysbiolab.eu

2 Installation requirements

Following software is required to run the Mfuzz package:

- R ($>$ 2.0.0). For installation of R, refer to *http://www.r-project.org*.
- R-package: e1071. For installation of these add-on packages, refer to *http://cran.r-project.org*.
- Bioconductor package: Biobase. To use the graphical user interface *Mfuzzgui*, the tcltk and Tkwidgets package needs to be installed. Refer to *http://www.bioconductor.org* for installation.

If these requirements are fulfilled, the *Mfuzz* add-on R-package can be installed. To see how to install add-on R-packages on your computer system, start *R* and type in *help(INSTALL)*. Optionally, you may use the R-function *install.packages()*. Once the *Mfuzz* package is installed, you can load the package by

```
> library(Mfuzz)
```

3 Data pre-processing

To illustrate our approach, we apply soft clustering to yeast cell cycle expression data by Cho *et al.* [3]. 6178 genes were monitored at 17 time points over a span of 160 minutes using

Affymetrix chips. Note that we do not exclude here the array corresponding to the time-point of 90 mins which displays irregularities in the expression values measured. Additionally, the data set was modified by restricting the number of genes to 3000. Thus, results here might differ from those reported in reference [1].

```
>   data(yeast)
```

3.1 Missing values

As a first step, we exclude genes with more than 25% of the measurements missing. Note that missing values should be denoted by `NA` in the gene expression matrix.

```
> yeast.r <- filter.NA(yeast, thres=0.25)
```

```
49 genes excluded.
```

Fuzzy c-means like many other cluster algorithms, does not allow for missing values. Thus, we replace remaining missing values by the average values expression value of the corresponding gene.

```
> yeast.f <- fill.NA(yeast.r,mode="mean")
```

Alternatively (and recommended), the (weighted) k-nearest neighbour method can be used (`mode='knn'/'wknn'`). These methods perform usually favourable compared to the simple method above, but are computationally intensive.

3.2 Filtering

Most cluster analyses published include a filtering step to remove genes which are expressed at low levels or show only small changes in expression. Different filtering procedures have been proposed. A popular procedure is the setting of a minimum threshold for variation
Calculation the standard deviation shows, however, that the transition between low and high values for variation in gene expression is smooth and no particular cut-off point is indicated (figure 1).

```
>   tmp <- filter.std(yeast.f,min.std=0)
```

Thus, the value of a filtering threshold remains arbitrary. As no stringent filtering procedure currently exists, we avoided any prior filtering of gene data. This prevents the loss of genes that may be biologically important.

3.3 Standardisation

Since the clustering is performed in Euclidian space, the expression values of genes were standardised to have a mean value of zero and a standard deviation of one. This step ensures that vectors of genes with similar changes in expression are close in Euclidean space:

```
> yeast.s <- standardise(yeast.f)
```

Importantly, Mfuzz assumes that the given expression data are fully preprocessed including any data normalisation. The function *standardise* does not replace the normalisation step. Note the difference: Normalisation is carried out to make different samples comparable, while standardisation (in Mfuzz) is carried out to make transcripts (or genes or proteins) comparable.

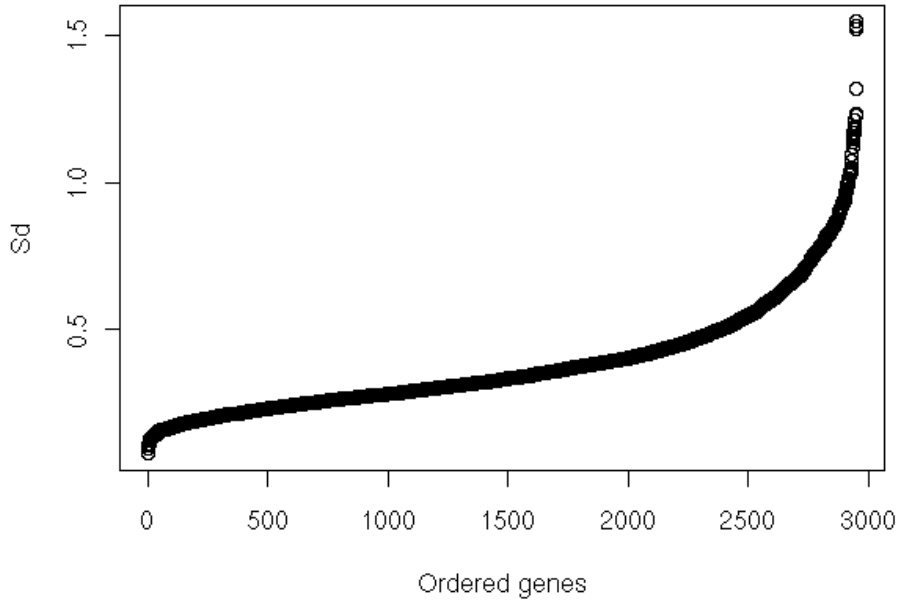

Figure 1: Standard deviation of gene expression vectors before standardisation. The genes were ordered by the standard deviation of the corresponding expression vector. A unique cut-off value for filtering is not prominent.

4 Soft clustering of gene expression data

Clustering is often applied to reveal regulatory mechanisms underlying gene expression. It is well known that regulation of genes is generally not in an 'on-off', but gradual manner which allows a finer control of the genes' functions. A cluster algorithm should reflect this finding by differentiating how closely a gene follows the dominant cluster patterns. Soft clustering appears as a good candidate for this task since it can assign a gene i gradual degrees of membership μ_{ij} to a cluster j. The membership values can vary continuously between zero and one. This feature enables soft clustering to provide more information about the structure of gene expression data.

Soft clustering is implemented in the function `mfuzz` using the fuzzy c-means algorithm (of the *e1071* package) based on minimization of a weighted square error function [4]. For soft clustering, the cluster centroids $\mathbf{c}_j$ result from the weighted sum of all cluster members and show the overall expression patterns of clusters. The membership values μ_{ij} indicate how well the gene i is represented by cluster $\mathbf{c}_j$. Low values μ_{ij} point to a poor representation of gene i by $\mathbf{c}_j$. Large values μ_{ij} point to a high correlation of the expression of gene i with the cluster centroid $\mathbf{c}_j$. The membership values are color-encoded in the plots generated by `mfuzz.plot`. This can facilitates the identification of temporal patterns in gene cluster (figure 2). You may also want to check out `mfuzz.plot2`, which works the same way as `mfuzz.plot`, but gives you a larger range of options, some of which are presented if you run `example(mfuzz.plot2)`.

Note that the clustering is based soley 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 averagered prior to clustering or placed into different distinct `ExpressionSet` objects.

```
> cl <- mfuzz(yeast.s,c=16,m=1.25)
> mfuzz.plot(yeast.s,cl=cl,mfrow=c(4,4),time.labels=seq(0,160,10))
```

4.1 Setting of parameters for FCM clustering

For fuzzy c-means clustering, the fuzzifier m and the number of clusters c has to be chosen in advance. For fuzzifier m, we would like to choose a value which prevents clustering of random data. Note, that fuzzy clustering can be tuned in such manner, that random data is not clustered. This is a clear advantage to hard clustering (such as *k-means*), which commonly detects clusters even in random data. To achieve this, different options exists: Either the function `partcoef` can be used to test, whether random data is clustered for a particular setting of m (see example of `partcoef`) or a direct estimate can be achieved using a relation proposed by Schwaemmle and Jensen [5]:

```
> m1 <- mestimate(yeast.s)
> m1 # 1.15
```

Setting of an optimal number of cluster c is usually challanging, especially for short time series and in case of overlapping clusters. Here, a range of c can be testedand a maximal value of c can be set which leads to the appearance of empty clusters (see `cselection`). Also, the minimum distance *Dmin* between cluster centroid can be utilized as an cluster validity index.

Here, we can monitor *Dmin* across a range of c [5]. We would expect that *Dmin* declines slower after reaching an optimal c (see also example of function `Dmin`). An alternativee way is to perform clustering with a range of cluster numbers and subsequently choose the optimal cluster number based on the assessment of their biological relevance e.g. by GO analyses.

4.2 Cluster cores

Membership values can also indicate the similarity of vectors to each other. If two gene expression vectors have a high membership value for a specific cluster, they are generally similar to each other. This is the basis for the definition of the core of a cluster. We define that genes with membership values larger than a chosen threshold α belong to the α-*core* of the cluster. This allows us to define relationships between genes within a cluster. Similarly to hierarchical clustering, the internal structures of clusters become accessible.

The average within-cluster variation is considerably reduced setting $\alpha = 0.7$. The use of the α-threshold can therefore act as *a posteriori* filtering of genes. This contrasts with previously discussed procedures which demand the problematic setting of a threshold *a priori* to the cluster analysis To extract list of genes belonging to the cluster cores, the `acore` function can be used.

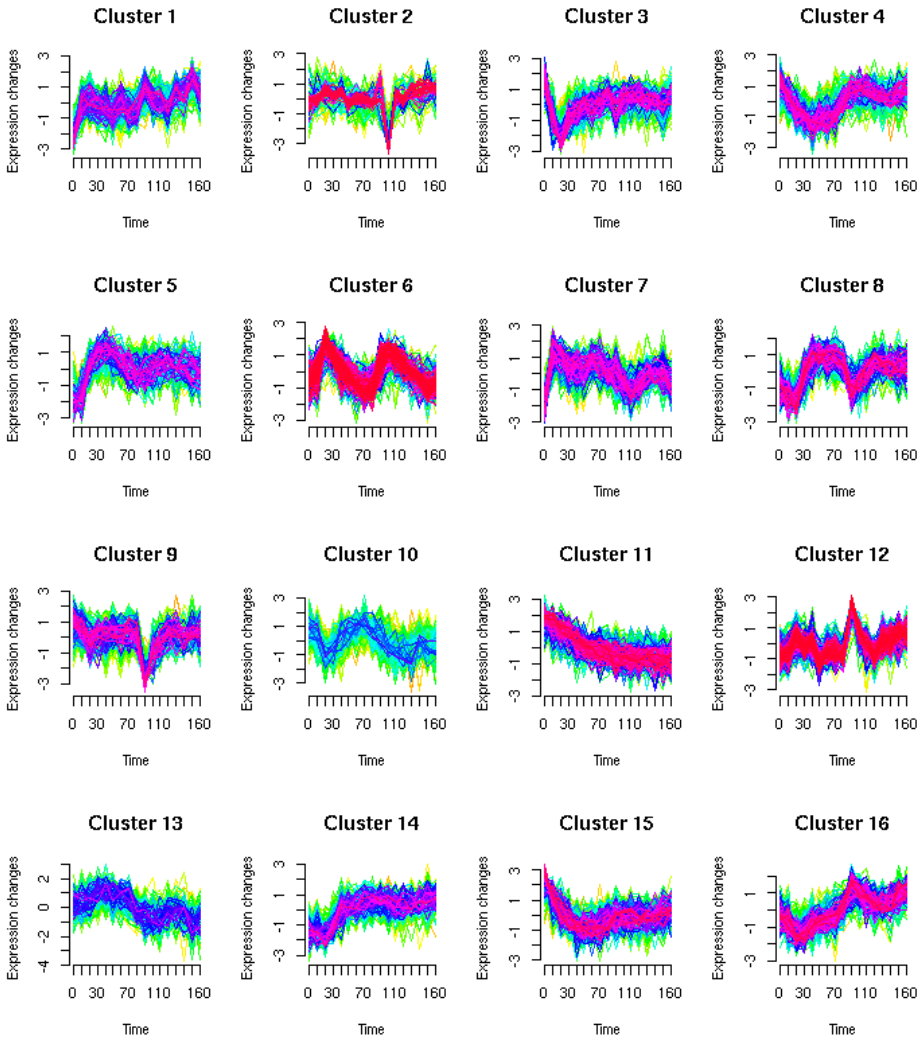

Figure 2: Soft clusters of yeast cell cycle expression data. Yellow or green colored lines correspond to genes with low membership value; red and purple colored lines correspond to genes with high membership value. Note the peaks in negative and positive direction for some clusters at time-point 90 min which may correspond to an experimental artifact.

5 Cluster stability

Variation of the FCM parameter m also allows investigation of the stability of clusters. We define here stable clusters as clusters that show only minor changes in their structure with variation of the parameter m. Stable clusters are generally well distinct of other clusters and compact. This is contrasted by weak clusters that lose their internal structure or disappear if m was increased.

```
>    cl2 <- mfuzz(yeast.s,c=16,m=1.35)
>    mfuzz.plot(yeast.s,cl=cl2,mfrow=c(4,4),time.labels=seq(0,160,10))
```

6 Global clustering structures

An interesting feature of soft clustering is the overlap or coupling between clusters. The coupling coefficient V_{kl} between cluster k and cluster l can be defined by

$$V_{kl} = \frac{1}{N} \sum_{i=1}^{N} \mu_{ik} \mu_{il} \tag{1}$$

where N is the total number of gene expression vectors. The coupling indicates how many genes are shared by two clusters. Clusters which have a low coupling show distinct overall patterns. If the coupling is large, clusters patterns are more similar. Hence, the coupling defines a similarity measure for pairs of clusters.

```
> O <- overlap(cl)
> Ptmp <- overlap.plot(cl,over=O,thres=0.05)
```

This allows the analysis of global clustering structures obtained by soft clustering, since relationships between clusters are defined. Similarly to hierarchical clustering, the global clustering structure can be examined at different resolutions determined by the cluster number c. For a small c, only the major clusters present in the data are obtained.

```
> cl3 <- mfuzz(yeast.s,c=10,m=1.25)
> mfuzz.plot(yeast.s,cl=cl3,mfrow=c(3,4))
> O3 <- overlap(cl3)
> overlap.plot(cl3,over=O3,P=Ptmp,thres=0.05)
```

If c is increased, sub-clusters with distinct patterns emerge Sub-clusters derived from a major cluster are generally strongly coupled, since they share the overall expression pattern. Finally, soft clustering produces empty clusters for further increase of c.

```
> cl4 <- mfuzz(yeast.s,c=25,m=1.25)
> mfuzz.plot(yeast.s,cl=cl4,mfrow=c(5,5))
> O4 <- overlap(cl4)
> overlap.plot(cl4,over=O4,P=Ptmp,thres=0.05)
```

7 Mfuzzgui - the graphical user interface for the Mfuzz package

The *Mfuzzgui()* function provides a graphical user interface for pre-processing, cluster analysis and visualization of gene expression microarray data using the functions of the *Mfuzz* package. The graphical interface is based on Tk widgets using the R-Tk interface developed by Peter Dalgaard. It also employs some pre-made widgets from the Bioconductor-package tkWidgets by Jianhua Zhang for the selection of objects or 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 assumes no existence of any exprSet objects and these objects can easily be constructed using the Mfuzzgui package itself from tab-delimited text files containing the gene expression data.

To start the graphical interface, type:

```
> Mfuzzgui()
```

A TclTk-Widget is launched (fig. 3). The use of the interface is intuitive. The GUI is divided into five sections; the order of these sections reflects the order of a standard analysis and visualisation procedure for a microarray data set:

1. Loading of data set

2. Pre-processing of the data set

3. Clustering

4. Cluster Analysis

5. Visualisation of Results of Clustering and Cluster Analysis

Note that only one data set can be analysed at a time. To visualize the results of various clustering methods and cluster analysis, it is not necessary to save/export this data set and reload it.

The general structure of the GUI is straight-forward. Each button corresponds to or is very similar to the name of the corresponding function in the Mfuzz package. If additional arguments for the function are needed, an input window is launched. Note that not everything will be checked to be of correct type. While some checks are implemented (e.g. checks if the loaded data set has the correct class), most arguments remains unchecked before the underlying function is called (e.g. it will not be checked if the array index is a positive integer.) If errors are produced, the validity of the input arguments should be examined.

For details about the required types of arguments and corresponding functions, please refer to the help pages of the Mfuzz package. In the following section, a brief introduction to the functionality of Mfuzzgui is given. As an example, the dataset *yeast* can be loaded in the global environment by

```
> data(yeast)
```

or alternatively can be loaded for analysis using the Browse objects buttons.

7.1 Loading data

exprSet object

- Browse objects Import of an exprSet object of class *exprSet* from the global environment (*.GlobalEnv*). A window produced by the function `objectBrowser` of the tkWidgets package is generated. Note that, although several objects can be selected, only the first selected object will be loaded. A check is performed if the object to be loaded belongs to the correct classes.

- Browse files Loading of an R data set of class *exprSet* stored in a file. A window produced by the function `fileBrowser` of the tkWidgets package is generated. Note that, although several files can be selected, only the first selected file is loaded. A check is performed if the object to be loaded belongs to the correct classes.

- Load table Construction and loading of an exprSet object from a tab-delimited data file. A window produced by the function `fileBrowser` of the tkWidgets package is generated. It is user's responsibility to use a correctly formated file (see also figure 4). The first row of the file contains sample labels and optionally, the second column can contains the time points. If the second row is used for the input the time, the first field in the second row must contain "Time". Similarly, the first column contains unique gene IDs and optionally second row can contain gene names. If the second row contains gene names, the second field in the first row must contain "Gene.Name". The rest of the file contains expression data. As example, two tables with expression data are provided.

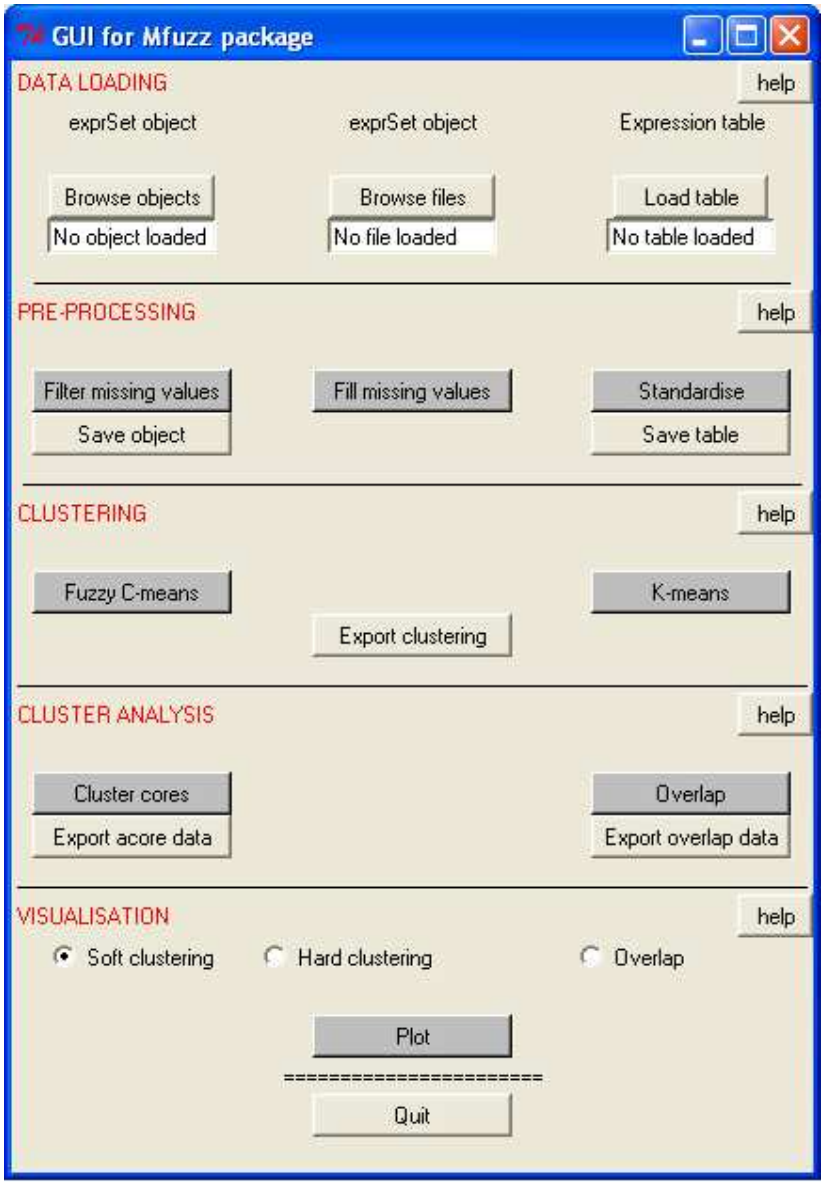

Figure 3: Screenshot of Mfuzzgui window

	A	B	C	D	E	F
1	GENE.ID	GENE.NAME	cdc28_0	cdc28_10	cdc28_20	cdc28_30
2	TIME		0	10	20	30
3	YDR132C		0.19	0.3	-0.29	0.29
4	YMR012W	CLU1	-0.15	-0.15	-0.04	-0.28
5	YLR214W	FERRIC REDUCTASE	0.38	0.3	-0.68	-0.52
6	YLR116W		0.17	0.06	-0.21	0.19
7	YDR203W		0.85	-0.1	-0.56	-0.31
8	YEL059C-A		0.45	0.2	0.06	0.1
9	YHR046C		0.37	0.31	0.07	-0.14
10	YJL053W	PEP8	0.38	-0.14	-0.14	0.17
11	YBL021C		-0.25	-0.09	0.34	-0.01
12	YHR034C		0.89	-0.02	-0.22	0.02
13	YMR167W	MutL Homolog	0.15	-0.26	-0.39	-0.68

Figure 4: Format of table for direct uploading. The row and column highlighted by yellow are optinal and their content is not utilized for clustering. Thus, they can potentially contain information other than the time of measurments or the gene name. However, the labels of these columns have to be as indicated in order to be recognized by Mfuzzgui.

These examples can be view by inputing `data(yeast.table)` and `data(yeast.table2)` in the R console.

The name of the current exprSet object is shown in the text fields below the buttons once they are loaded into the current environment.

7.2 Pre-processing

- Filter missing values: This button performs two tasks. First it generates a pop-up window for the selection of threshold value for filtering the genes. Then after pressing the *OK* button of the pop-up window, it removes the genes which have more NAs (missing values) than the specified threshold value.

- Fill missing values Like before, this button again accomplishes two tasks. First it produces a pop-up window asking for the parameter values and method to be used for the replacement of missing values. After pressing the *OK* button, it replaces the missing values present in the partially processed data set using the user specified method and parameter values.

- Standardise Call of the function `standardise`. This step will standardise the gene expression values so that they have a mean value of zero and a standard deviation of one.

- Save object This button can be used to save the pre-processed object to some R data file. A window is generated to enable the user specify the file name where the object is

to be stored.

- Save table This button can be used to save the pre-processed data in a tabular form in some disk file. A pop-up window is generated to facilitate user to specify the file name which can later be used to reload the data in the form of an exprSet object.

7.3 Clustering

- Fuzzy C-means: Call of function `mfuzz`. This button performs the fuzzy c-means clustering of the data. A new window is generated so that user can specify the parameter values which are to be used in clustering the data.

- K-means Call of the function `kmeans2`. This button can be used for performing the standard k-means clustering. The parameter values can be specified in the pop-up window.

- Export clustering This button can be used to save the results of clustering into a text file. In the case of fuzzy C-means method, it stores the membership values of each gene for each cluster in a tabular format whereas for the k-means method, it stores the cluster vector containing the cluster numbers corresponding to each gene.

Note that the clustering is based soley 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 averagered prior to clustering or placed into different distinct `ExpressionSet` objects. Similarly, if a table has been uploaded, the ordering of the samples is based on the ordering of the columns and not on the time of measurements given in the optinal second row.

7.4 Cluster Analysis

- Cluster cores Call to the function `acore`.This function extracts genes forming the alpha cores of soft clusters. The minimum membership value can be specified in the pop-up window.

- Overlap Call to the function `overlap`. This function calculates the overlap of clusters produced by *mfuzz*.

- Export acore data This button exports the results produced by *Cluster cores* button in a text file.

- Export overlap data This button, like the previous one also saves the results generated by *Overlap* button into a text file. To see the details please refer to the documentation of *Mfuzz* package.

7.5 Visualisation

- Soft clustering Call to the function `mfuzz.plot`. This button can be used to visualise the results of soft clustering in the form of color-coded plots. Some parameters can be specified in the pop-up window.

- Hard Clustering Call to the function `kmeans2.plot`. This function visualises the clusters produced by *mfuzz*.

- Overlap Call to the function `overlap.plot`. This function visualises the cluster overlap produced by `Overlap`.

7.6 Help

Apart from the documentations provided with the *Mfuzzgui* package, there are *help* buttons in each section of the *Mfuzzgui* GUI. When one clicks on a help button, it gives a brief overview in a message box of what each button in the corresponding section does.

References

[1] M.E. Futschik and B. Charlisle, Noise robust clustering of gene expression time-course data, Journal of Bioinformatics and Computational Biology, Vol. 3, No. 4, 965-988, 2005

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

[3] Cho RJ, Campbell MJ, Winzeler EA, Steinmetz L, Conway A, Wodicka L, Wolfsberg TG, Gabrielian AE, Landsman D, Lockhart DJ, Davis RW, A genome-wide transcriptional analysis of the mitotic cell cycle, *Mol Cell*, **2**:65–73, 1998

[4] Bezdak JC, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981

[5] Schwaemmle and Jensen, Bioinformatics,Vol. 26 (22), 2841-2848, 2010