BIOINFORMATICS APPLICATIONS NOTE Vol. 19 no. 5 2003, pages 659–660
DOI: 10.1093/bioinformatics/btg046

MatArray: a Matlab toolbox for microarray data

David Venet [1, 2]

[1]I.R.I.B.H.M. - Campus Hopital Erasme, Route de Lennik 808 Bat C, CP 602-1070 Brussels, Belgium and [2]I.R.I.D.I.A. - Université Libre de Bruxelles, 50, Av. F. Roosevelt, CP 194/6-1050 Brussels, Belgium

Received on August 19, 2002; revised on October 29, 2002; accepted on November 4, 2002

ABSTRACT

Summary: The microarray technology allows the high-throughput quantification of the mRNA level of thousands of genes under dozens of conditions, generating a wealth of data which must be analyzed using some form of computational means. A popular framework for such analysis is Matlab, a powerful computing language for which many functions have been written. However, although complex topics like neural networks or principal component analysis are freely available in Matlab, functions to perform more basic tasks like data normalization or hierarchical clustering in an efficient manner are not. The MatArray toolbox aims at filling this gap by offering efficient implementations of the most needed functions for microarray analysis. The functions in the toolbox are command-line only, since it is geared toward seasoned Matlab users.

Availability: http://www.ulb.ac.be/medecine/iribhm/microarray/toolbox

Contact: davenet@ulb.ac.be

INTRODUCTION

The microarray technology allows the high-throughput quantification of the mRNA level of thousands of genes under dozens of conditions, generating of wealth of data which must be analyzed using some form of computational means. However, most of the programs available are either closed black-box tools, in which a series of algorithms are hardwired, or completely open architecture like Matlab (MathWorks inc.) or R (Ihaka and Gentleman, 1996). There is a need for something in between, offering at the same time the flexibility of an open architecture and the algorithm comprehensiveness of a package. Such a need has been fulfilled for the R community, where a few packages of functions designed specifically for microarrays have been developed (e.g. BioConductor). We propose here a toolbox to similarly complete the Matlab environment with the functions most commonly needed when dealing with microarrays.

The Matlab environment presents many advantages for the analysis of microarray data: it offers an efficient and natural way of dealing with large data sets, it provides a comprehensive set of functions, and some of the complex tools used in microarrays, like principal components analysis, are built-in. Moreover, many toolboxes have been developed and are often freely available for more specialized needs, like for instance neural networks or support vector machines. However, quite surprisingly, more basic tasks like hierarchical clustering or non-linear normalization are not available, or only in a very inefficient implementation.

The toolbox presented here should fulfill most obvious needs. The main tools present are: normalization procedures, including the loess-type non-linear normalization, a fast implementation of the hierarchical and K-means clustering, a possibility to export the clustering to TreeView for display and the computation of the optimal ordering of the nodes in a clustering.

FEATURES

The toolbox provides functions to deal with the data after the slide quantification process, for which specialized tools already exist.

The first step after the quantification is to load the data in Matlab. This is not a problem if the data are presented as a rectangular numerical table, but the outputs of the quantification programs are usually more complex than that. A script in the toolbox allows the loading of the files produced by ImaGene (Biodiscovery inc.) directly. This script can be modified to take other formats into account.

After the files have been loaded, the values should be normalized. Normalization has long been viewed simply as a scaling of the values in order to correct for the variation in the laser intensity or the amount of material on the slide. However, it has appeared that other systematic errors are often present and that they can be removed with a sensible data normalization (Tseng *et al.*, 2001; Workman *et al.*, 2002; Yang *et al.*, 2002). Two main sources of error have been presented in the literature: a geometrical effect, that is a dependence between the intensity or the red/green ratio of a spot and its location

on the slide, and an intensity effect, that is a dependence of the red/green ratio of a spot on its intensity. Two normalization functions are available in the toolbox to remove those two types of systematic error. The first corrects for the geometrical effect by normalizing the spots in a certain neighborhood on the slide. The second corrects the intensity dependence by evaluating its effect with an implementation of loess (Cleveland and Grosse, 1991), and then applying a correction as described in Yang *et al.*

Once the data have been properly loaded and normalized, they must be analyzed. One of the classical ways to perform such analysis is to use a clustering algorithm (Eisen *et al.*, 1998; Lukashin and Fuchs, 2001; Getz *et al.*, 2000), the most common one being the hierarchical clustering. Such algorithms are indeed present in Matlab (in the Statistic Toolbox), but are extremely inefficient. A mex-file, that is a routine coded in C callable from Matlab, has been programmed instead. This leads to a much faster implementation. For instance, the average linkage clustering of 2500 genes on a typical workstation using the toolbox takes about 6 seconds. By comparison, the program Cluster from Michael Eisen takes about 22 seconds and the implementation offered in the Statistic toolbox was still running after an hour. Single linkage, average linkage, complete linkage and Ward's method hierarchical clustering are available.

Two functions are offered to translate the clusterings obtained to the format of the Statistic Toolbox and back, so the tools it offers can be used. However, those tools are not very adapted for the visualization of microarray data. For this reason, a function has been created which allows the export of the clusterings to a format which can be read by TreeView, the program developed by Michael Eisen (Eisen *et al.*, 1998).

A hierarchical clustering can always be transformed by permuting the left and right branch at any node. A natural way to decide which clustering is better is to measure the sum of the distances between adjacent leaves. An algorithm which computes the optimal clustering has been proposed by Bar-Joseph *et al.* (2001). A mex-file version of this algorithm has been created, so that it can be called from Matlab.

Another clustering algorithm which is often used is K-means, although its neural network form, the self-organizing map, is probably more popular. Self-organizing map toolboxes are available, but the K-means itself is curiously missing from the Matlab toolboxes. Hence, a straightforward K-means implementation is offered, as well as a modified version which often improves the quality of the solutions. The regular K-means present is about seven times faster than the batch implementation of a freely available SOM toolbox (http://www.cis.hut.fi/projects/somtoolbox/), while providing similar results.

In conclusion, the functions of the MatArray toolbox should offer the most needed tools to any researcher who wants to handle microarray data using the Matlab environment. This should permit focus on interesting problems, instead of reprogramming the same functions over and over again.

The toolbox is offered free of charge to all users on the website (http://www.ulb.ac.be/medecine/iribhm/microarray/toolbox).

ACKNOWLEDGEMENTS

This work was supported by the Région Wallonne and UCB Pharma. We thank Ziv Bar-Joseph for his code of the optimal leaf ordering algorithm.

REFERENCES

Bar-Joseph,Z., Gifford,D.K. and Jaakkola,T.S. (2001) Fast optimal leaf ordering for hierarchical clustering *Proc. 9th ISMB. Bioinformatics*, **17**, S22–S29.

Cleveland,W.S. and Grosse,E. (1991) Computational methods for local regression. *Statistics and Computing*, **1**, 47–62.

Eisen,M.B., Spellman,P.T., Brown,P.O. and Botstein,D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci. USA*, **95**, 14863–14868.

Getz,G., Levine,E., Domany,E. and Zhang,Q. (2000) Superparamagnetic clustering of yeast gene expression profiles. *Physica A*, **279**, 457–464.

Ihaka,R. and Gentleman,R. (1996) A language for data analysis and graphics. *J. Comput. Graphical Statist.*, **5**, 299–314.

Lukashin,A.V. and Fuchs,R. (2001) Analysis of temporal gene expression profiles: clustering by simulated annealing and determining the optimal number of clusters. *Bioinformatics*, **17**, 405–414.

Tseng,G.C., Oh,M.K., Rohlin,L., Liao,J.C. and Wong,W.H. (2001) Issues in cDNA microarray analysis: quality filtering, channel normalization, models of variations and assessment of gene effects. *Nucleic Acid Res.*, **29**, 2549–2557.

Workman,C., Jensen,L.J., Jarmer,H., Berka,R., Gautier,L., Nielsen,H.B., Saxild,H.-H., Nielsen,C., Brunak,S. and Knudsen,S. (2002) A new non-linear normalization method for reducing variability in DNA microarray experiments. *Genome Biol.*, **3**, research0048.1–0048.16.

Yang,Y.H., Dudoit,S., Luu,P., Lin,D.M., Peng,V., Ngai,J. and Speed,T.P. (2002) Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acid Res.*, **30**, e15.