MatArray: a Matlab toolbox for microarray data

David Venet\textsuperscript{1, 2}

\textsuperscript{1}I.R.I.B.H.M. - Campus Hopital Erasme, Route de Lennik 808 Bat C, CP 602-1070 Brussels, Belgium and \textsuperscript{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 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 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.

The toolbox presents 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 ImeGene (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 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/riibhm/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


