BIOINFORMATICS APPLICATIONS NOTE

Vol. 26 no. 16 2010, pages 2062–2063
doi:10.1093/bioinformatics/btq334

Gene expression

Advance Access publication July 29, 2010

ExpressionView—an interactive viewer for modules identified in

Andreas Lüscher[1,†], Gábor Csárdi[1,2,†], Aitana Morton de Lachapelle[1,2,†], Zoltán Kutalik[1,2,†], Bastian Peter[1,2] and Sven Bergmann[1,2,*]

[1]Swiss Institute of Bioinformatics and [2]Department of Medical Genetics, University of Lausanne, Lausanne, Switzerland

Associate Editor: Martin Bishop

ABSTRACT

Summary: ExpressionView is an R package that provides an interactive graphical environment to explore transcription modules identified in gene expression data. A sophisticated ordering algorithm is used to present the modules with the expression in a visually appealing layout that provides an intuitive summary of the results. From this overview, the user can select individual modules and access biologically relevant metadata associated with them.

Availability: http://www.unil.ch/cbg/ExpressionView. Screenshots, tutorials and sample data sets can be found on the ExpressionView web site.

Contact: sven.bergmann@unil.ch

Received on February 2, 2010; revised on May 31, 2010; accepted on June 21, 2010

1 INTRODUCTION

Biclustering is an unsupervised data analysis method that is frequently used to explore microarray data. Biclustering algorithms process collections of expression profiles to identify groups of genes co-expressed under some conditions (samples). We refer to such groups as modules. While there is a multitude of biclustering software available [for reviews and comparisons see (Ihmels and Bergmann, 2004; Madeira and Oliveira, 2004; Prelić *et al.*, 2006)], packages with intuitive interfaces that allow for an interactive exploration of the results are sparse.

Existing approaches include Bivisu (Cheng *et al.*, 2007) and BicOverlapper (Santamaría *et al.*, 2008). The former is an interactive biclustering program that plots modules individually, making it difficult to identify the relationship between the overlapping modules. The latter is a novel tool for plotting overlapping modules, yet in an abstract space. Our approach in ExpressionView is different, as we use the usual gene-sample space and visualize all modules together, on top of the reordered expression matrix. The reordering ensures that the genes and samples that appear in the same module are kept together.

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first four authors should be regarded as joint First authors.

2 PACKAGE DESIGN AND WORKFLOW

With the ExpressionView package, bicluster analysis can be separated into two parts. The first part involves finding the modules in the dataset with some algorithm, possibly running enrichment analysis for the modules, and reordering the rows and columns of the expression matrix according to the modules. This part is typically done by a bioinformatician. The second part of the analysis involves the visualization and interactive exploration of the results. This part is typically done by researchers without extensive programming knowledge.

The first part of ExpressionView is written in GNU R (R Development Core Team, 2009) and contains an implementation of the matrix reordering algorithm. The second part is an interactive visualization tool in the form of an Adobe Flash applet, for which the user only needs a Flash-enabled web browser.

This dual implementation has the advantage that all the power of the GNU R environment and the BioConductor (Gentleman *et al.*, 2004) packages can be used for the analysis itself, e.g. all organisms that are (and will be) supported by BioConductor are automatically supported by ExpressionView. On the other hand, the exploration of the results does not need any GNU R knowledge and in most cases no extra software needs to be installed. See Figure 1 for a typical ExpressionView workflow.

2.1 Reordering genes and conditions

ExpressionView is designed to work with gene expression data in the form of a Bioconductor *ExpressionSet*. This class provides a user-friendly way to access the actual gene expression matrix and its associated metadata. ExpressionView can treat biclustering results obtained by the Iterative Signature Algorithm (Bergmann *et al.*, 2003; Csárdi *et al.*, 2010) and any of the methods available in the Biclust package (Kaiser and Leisch, 2008). Since the structure of biclustering results is independent of the algorithm, an extension to other methods is straightforward.

To present the collection of possibly overlapping modules in a visually appealing form, it is necessary to reorder the rows (conditions) and columns (genes) of the gene expression matrix in such a way that biclusters form contiguous rectangles. Since it is in general impossible to find such an arrangement for more than two mutually overlapping modules, we propose here an approximate solution that optimizes the arrangement within the original data, by maximizing the total area of the largest contiguous module subsets. [An alternative would be to repeat rows and columns as

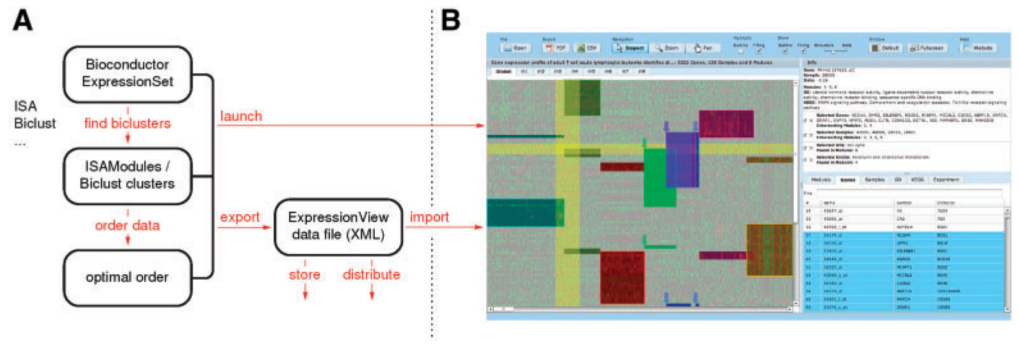

Fig. 1. Workflow of ExpressionView, showing the two parts of the analysis. **(A)** These steps are performed by a bioinformatician, using GNU R. Starting from gene expression data in the form of a Bioconductor *ExpressionSet*, the first step is finding the modules. In a second step, the rows and columns of the gene expression matrix are rearranged to produce an easily readable overview of the results. The last step consists of combining the gene expression data and its associated metadata, possibly including results of enrichment analysis, with the results from the biclustering and produce an ExpressionView data file. **(B)** This file can be distributed and finally explored with the interactive Flash applet by the end user. Please see the web site for details on the data file format.

necessary (Grothaus *et al.*, 2006), but for many modules this results in a very large expression matrix.]

This optimization task is an interesting problem on its own, which to the best of our knowledge has not been studied in the literature. We briefly outline our strategy here (see our web site for details). The reordering of the rows is independent from that of the columns, so the same optimization method can be applied separately to rows and columns. For a given order of the elements (either genes or conditions), we compute for each module i the size of the largest contiguous sequence of elements (i.e. the maximal number of neighboring elements $N_i^{\max}$). Then, as a measure of the quality (Q) of the order, we sum this quantity over all modules ($Q=\sum_i N_i^{\max}$). To optimize Q, an initial sequence is calculated using hierarchical clustering. Two operations are then applied to this: (i) permutations that exchange two elements within a module and (ii) shifts of a sequence of multiple elements of the same module to a different position. We use a greedy iterative scheme that performs these operations at all possible positions and keeps the best new sequence if it improves Q. The algorithm stops if after a given number of operations no significant improvement of Q is achieved.

We have studied a large number of perfectly orderable, but initially scrambled, test cases. We find that the proposed algorithm finds an order that recovers >99% of the score of the optimal solution and in most cases, it recovers the correct alignment. For random samples, which are more representative for actual gene expression data, the execution time increases polynomially with the number of clusters m as $\mathcal{O}(m^{\alpha})$, where $\alpha \in [1.6, 2]$, almost independently of the number of elements n. For a given number of clusters, we find $\mathcal{O}(n^{\alpha})$, with $\alpha \in [2.5, 2.7]$.

Once the optimal order is determined, the program rearranges the gene expression matrix accordingly and exports all the relevant information into an XML file, that can be placed on a web server or distributed by email and then imported by the interactive viewer.

2.2 Visualization

A screenshot of the viewer is shown in Figure 1. The interface is divided in two parts: on the left-hand side, the user finds the gene expression data in the common heat map form, on top of which the modules are overlaid. On the right-hand side, the metadata associated with the expression data and the results of the enrichment calculations for GO (Ashburner *et al.*, 2000) categories and KEGG (Kanehisa *et al.*, 2004) pathways are shown. Wherever possible, these elements are linked to the corresponding databases. The interface essentially behaves as an image viewer, allowing the user to zoom and pan around the expression data, getting instant feedback on the selected item.

Funding: Swiss Institute of Bioinformatics, the Swiss National Science Foundation (3100AO-116323/1); European Framework Project 6 (through the EuroDia and AnEuploidy projects).

Conflict of Interest: none declared.

REFERENCES

Ashburner,M. *et al.* (2000) Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat. Genet.*, **25**, 25–29.

Bergmann,S. *et al.* (2003) Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys. Rev. E*, **67**, 031902.

Cheng,K. *et al.* (2007) BiVisu: software tool for bicluster detection and visualization. *Bioinformatics*, **23**, 2342–2344.

Csárdi,G. *et al.* (2010) Modular analysis of gene expression data with R. *Bioinformatics*, **26**, 1376–1377.

Gentleman,R.C. *et al.* (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, **5**, R80.

Grothaus,G. *et al.* (2006) Automatic layout and visualization of biclusters. *Algorithms Mol. Biol.*, **1**, 15.

Ihmels,J.H. and Bergmann,S. (2004) Challenges and prospects in the analysis of large-scale gene expression data. *Brief. Bioinform.*, **5**, 313–327.

Kaiser,S. and Leisch,F. (2008) A toolbox for bicluster analysis in R. In Brito,P. ed. *Compstat 2008-Proceedings in Computational Statistics*, University of Munich, Physica, Heidelberg, Germany, pp. 201–208.

Kanehisa,M. *et al.* (2004) The kegg resource for deciphering the genome. *Nucleic Acids Res.*, **32** (Database issue), 277–280.

Madeira,S. and Oliveira,A. (2004) Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **1**, 24–45.

Prelić,A. *et al.* (2006) A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, **22**, 1122–1129.

R Development Core Team (2009) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Santamaría,R. *et al.* (2008) BicOverlapper: a tool for bicluster visualization. *Bioinformatics*, **24**, 1212–1213.