BIOINFORMATICS

Vol. 00 no. 00 2009
Pages 1–3

Modular analysis of gene expression data with R

Gábor Csárdi ,[1,2], Zoltán Kutalik [1,2] and Sven Bergmann [1,2]

[1]Department of Medical Genetics, and [2]Swiss Institute of Bioinformatics, University of Lausanne, Rue de Bugnon 27, CH-1005 Lausanne, Switzerland.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Summary: Large sets of data, like expression profile from many samples, require analytic tools to reduce their complexity. Classical (bi-)clustering algorithms typically attribute elements (genes, arrays) to disjoint groups (”clusters”). Yet, in some cases overlapping cluster assignments would suit the biological reality much better.

The Iterative Signature Algorithm (ISA) was designed to overcome this and other limitations of standard clustering algorithms. It aims to reduce the complexity of very large sets of data by decomposing it into so-called ”modules”. In the context of gene expression data these modules consist of subsets of genes that exhibit a coherent expression profile only over a subset of microarray experiments. Genes and arrays may be attributed to multiple modules and the level of required coherence can be varied resulting in different ”resolutions” of the modular mapping. Since the ISA does not rely on the computation of correlation matrices (like many other tools), it is extremely fast even for very large datsets.

In this short note, we introduce two BioConductor Gentleman *et al.* (2004) software package written in GNU R: The ‘`isa2`’ includes a highly optimized implementation of the ISA and ‘`eisa`’ provide a convenient interface to run the ISA and visualize its output.

Availability: http://www.unil.ch/cbg/homepage/software.html

Contact: Sven.Bergmann@unil.ch

1 INTRODUCTION

The ISA can be applied to identify coherent substructures (i.e. modules) from any rectangular matrix of data. To be specific we consider here the case of transcriptomics data corresponding to a selection of gene-expression profiles from a collection of samples. The method has been described in detail in Refs **??**. Here only give a brief summary:

The ISA identifies modules by an iterative procedure. The algorithm starts from an input seed (corresponding to some set of genes or samples) which are refined at each iteration by adding and/or removing genes and/or samples until the process converges to a stable set, which is referred to as a transcription module. This iterative procedure is typically performed to a large number of seeds. In the unsupervised approach these seeds are chosen randomly to sample uniformly the immense search space. We also implemented a semi-supervised method, to which we refer as ”smart seeding”, where the seeds a biased to start with certain sets of genes or samples for which there is prior knowledge.

The output of ISA is a collection of potentially overlapping modules. Each gene and each sample is attributed a score that reflects the strength of the association with the module.

In the following we first outline how our software tool is commonly for a modular analysis (Section 2) and then we describe in more detail the R packages that implement the ISA and visualize its output (Section 3).

2 METHODS

A typical modular analysis for gene expression data includes the following steps:

Batch correction. In order study the global organization of a transcription program including many aspects of transcriptional regulation one often combines several microarray experiments into a single dataset. In such a case it is important to properly normalize such data and reduce the bias due to constituent datasets. Several methods address this challenge, see e.g. Johnson *et al.* (2007) for an algorithm that has a GNU R implementation. (S: Do we have a standard flag to call their software?)

Gene filtering. Genes that have very low expression levels in all samples, carry little if any information and may reflect ineffective array probes, etc. Since these genes are likely to contribute only noise to the analysis, we integrated the possibility to remove such genes. (S: Describe how this is done, and if it requires setting a threshold!)

ISA normalization. In each iteration of the ISA weighted sums of expression levels over either genes or samples are computed. for a number of . Since different genes typically show different levels of base expression and variance, it is important to standardize expression levels to Z-scores. The ISA uses two sets of Z-scores, one calculated for each gene across all samples, the other for each sample across all genes.

Random and smart seeding. As mentioned previously, ISA can be performed with random or smart seeds, based on the application.

The ISA iteration. This step is essentially performing the ISA and finding the biclusters from the starting seeds.

Merging the modules. It is possible that more seeds converge to the same, or very similar biclusters. This step eliminates such duplicates.

Robustness of biclusters. To access the significance of a bicluster, we designed a robustness measure, that can be used to filter out spurious modules. This is done based on performing ISA on the scrambled input data.

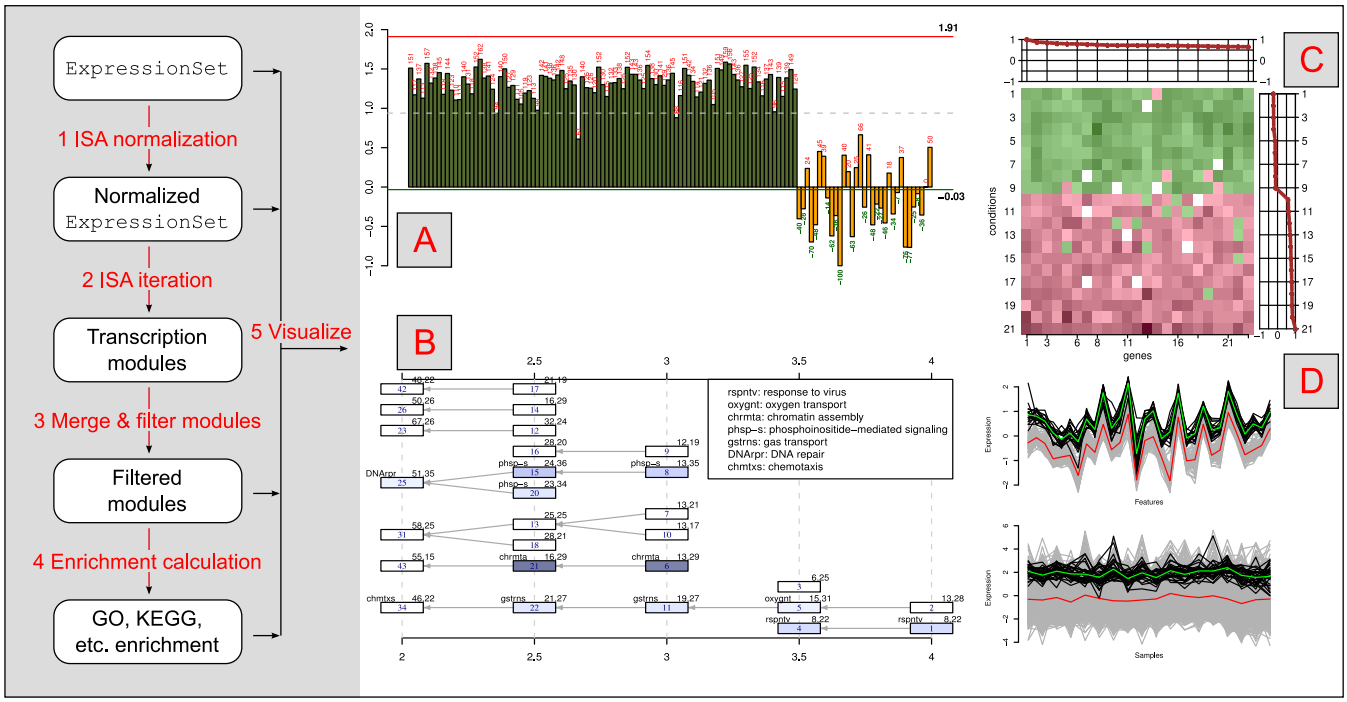

Fig. 1. Work flow of a typical modular analysis study, with the 'eisa' package (left); and four bicluster visualization techniques (right). The input of the 'eisa' pipeline is a BioConductor *ExpressionSet* object. This is normalized to Z-scores (1); the ISA iterations are initiated using "smart" or random seeds (2); and the biclusters are then merged and filtered based on the robustness measure (3). The 'eisa' package contains tools to perform many GO, KEGG, etc. enrichment calculation quickly (4), and several bicluster visualization tools (5). Subfigures **A**, **B**, **C** and **D** were generated using the acute lymphoblastic leukemia data set, see Chiaretti *et al.* (2004) and the 'ALL' R package. **(A)** A condition plot for a simple bicluster. Each sample is represented as a bar, its height is its score in the bicluster. Samples above the red (none in this case) and samples below the green line are part of the biclusters. The coloring indicates cases and controls. **(B)** A module tree. Each rectangle is a bicluster; see the definition of the edges in the text. The modules are colored according to their Gene Ontology enrichment p-values, the codes of the enriched GO categories are shown in the top-left corner of the rectangles. The top-right corner shows the number of genes and conditions in the bicluster. The gene thresholds used for finding the modules is shown on the horizontal axes. **(C)** Heatmap for a single module, showing correlated expression for the genes and samples. The red lines are the gene and sample scores. **(D)** Profile plot, for a single module. On the top plot the expression of all samples is plotted, the samples of the module with black. The red and green lines show the mean of the two groups. The bottom plot is the same for the genes of the module.

Module trees. The ISA works with two stringency threshold parameters, the gene threshold and the sample threshold. ISA modules can be organized into a directed graph, a module tree, in which there is an edge from module A to module B, if the ISA converges to module B from module A, with the same threshold parameters that were used to find module B. A module tree provides a hierarchical modular description of a data set.

3 IMPLEMENTATION

The ISA and accompanying visualization tools are implemented in two R packages. The 'isa2' package contains the implementation of the basic ISA itself; this package can be used to analyze any tabular data. The 'eisa' package builds on 'isa2'. It adds support to standard BioConductor gene expression data structures; and contains gene expression specific visualization tools, see some of these in Fig. 1.

Both the 'isa2' and 'eisa' packages support two workflows. The *simple workflow* involves a single R function call and runs all ISA steps (1-3 on Fig. 1.) with their default parameters.

The *detailed workflow* means running every step of the modular analysis separately, possibly with non-default parameters. This allows the users to tailor the ISA completely according their needs.

For users that prefer Matlab over R, we created a Matlab package for ISA, please see the ISA homepage for more information.

The 'eisa' package implements a set of visualization techniques for biclusters, some of these are specific to ISA, others are not. Please see Fig. 1 for some of these.

The 'ExpressionView' R package implements an interactive tool for visualizing ISA (or other) overlapping biclusters.

The 'biclust' package, Kaiser *et al.* (2009), implements a number of biclustering algorithms in a unified framework. The 'eisa' package include tools to convert between 'biclust' and ISA biclusters. This allows the cross-talk of the functions in the two packages.

ACKNOWLEDGEMENT

Funding: The authors are greatful to the Swiss Institute of Bioinformatics for support.

Conflict of interest: None declared.

REFERENCES

Chiaretti, S., Li, X., Gentleman, R., Vitale, A., Vignetti, M., Mandelli, F., Ritz, J., and Foa, R. (2004). Gene expression profile of adult t-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, **103**(7).

Gentleman, R. C., Carey, V. J., Bates, D. M., *et al.* (2004). Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, **5**, R80.

Johnson, W., Rabinovic, A., and Li, C. (2007). Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, **8**(1), 118–127.

Kaiser, S., Santamaria, R., Theron, R., Quintales, L., and Leisch., F. (2009). *biclust: BiCluster Algorithms*. R package version 0.8.1.