BIOINFORMATICS

Vol. 00 no. 00 2009
Pages 1–2

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 profiles 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 a very large set 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 datasets.

In this short note, we introduce two BioConductor (Gentleman *et al.*, 2004), software packages written in GNU R: The '`isa2`' includes a highly optimized implementation of the ISA and '`eisa`' provides a convenient interface to run the ISA, visualize its output and puts the biclusters into biological context.

Availability: `http://www.unil.ch/cbg/ISA`

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 set of gene-expression profiles from a collection of samples. The method has been described in detail in Refs (Ihmels *et al.*, 2004; Bergmann *et al.*, 2003). Here we 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 is 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.

The output of ISA is a collection of potentially overlapping modules. In every module, 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 used 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. To 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 additional data normalization is crucial to 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.

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 (Hackstadt and Hess, 2009), we suggest removing them before running the module identification of the ISA.

ISA normalization. (Step 1 on Fig. 1.) In each iteration the ISA computes thresholded weighted sums of expression levels over either genes or samples. 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, ISA iteration (Step 2.) The iterative procedure of the module identification is typically applied 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 are biased to start with certain sets of genes or samples based on prior knowledge. The ISA can be performed with random or smart seeds, depending on the application.

Merging and filtering the modules. (Step 3.) It is possible that several seeds converge to the same, or very similar biclusters. This step eliminates such duplicates. To access the significance of a module, we designed a robustness measure, that can be used to filter out spurious modules. This is done by applying the ISA to scrambled input data in order to obtain a reference (null) distribution for the significance scores.

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, to which we refer as "module tree". An edge from module A to module B indicates that 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 dataset.

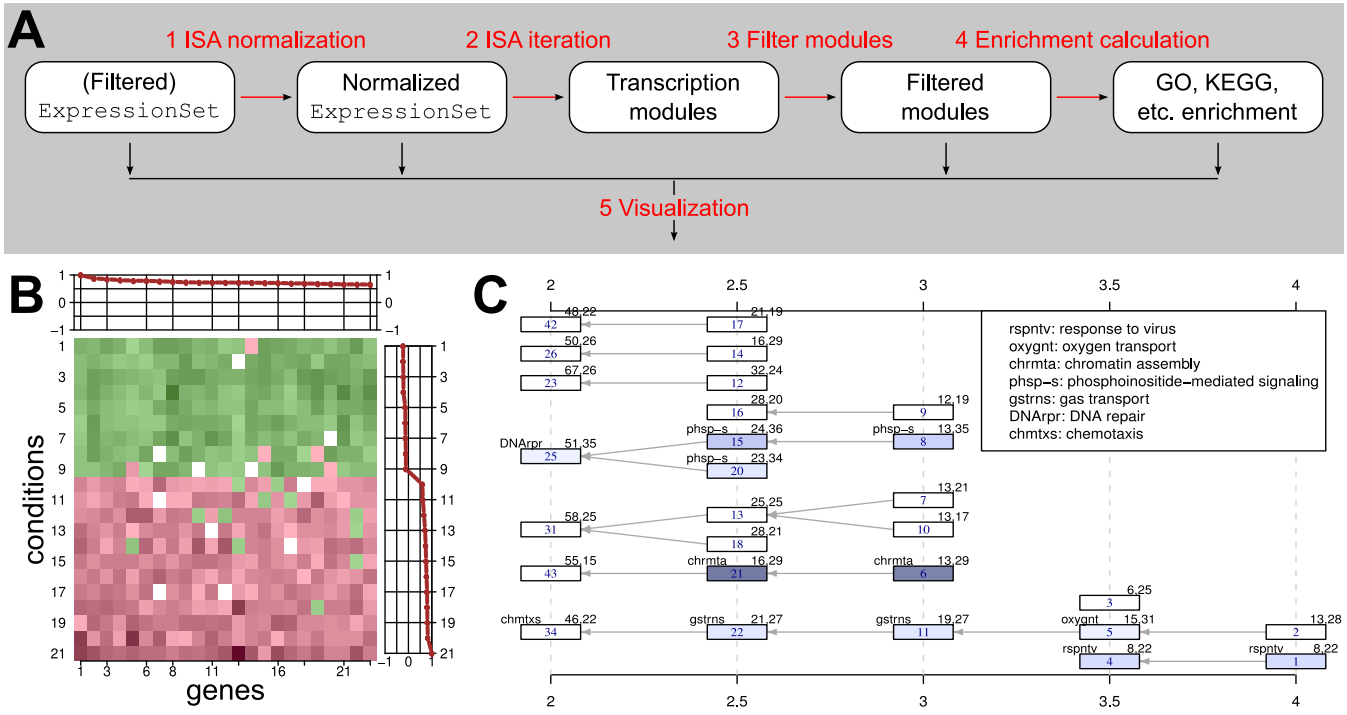

Fig. 1. **(A)** Work flow of a typical modular analysis with the 'eisa' package. See text for details. Subfigures **B** and **C** were generated using the acute lymphoblastic leukemia data set, see (Chiaretti *et al.*, 2004) and the 'ALL' R package. **(B)** Heatmap for a single module, showing coherent expression of the genes across the samples. The red lines are the gene and sample scores. **(C)** Module tree. Each rectangle is a module; see the definition of the edges in the text. 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 module. The gene thresholds used for finding the modules is shown on the horizontal axes.

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 data structures; and contains gene expression specific visualization tools (see Fig. 1 for examples).

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 in Fig. 1) with their default parameters.

In the *detailed workflow* every step of the modular analysis is executed separately, possibly with non-default parameters. This allows the users to tailor the ISA 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 modules (see Fig. 1 for examples).

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.

Additional information about the ISA, the 'isa2' and 'eisa' packages and several tutorials are available on the ISA homepage.

ACKNOWLEDGEMENT

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

Conflict of interest: None declared.

REFERENCES

Bergmann, S., Ihmels, J., and Barkai, N. (2003). Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys Rev E Nonlin Soft Matter Phys*, page 031902.

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.

Hackstadt, A. and Hess, A. (2009). Filtering for increased power for microarray data analysis. *BMC Bioinformatics*, **10**(1), 11.

Ihmels, J., Bergmann, S., and Barkai, N. (2004). Defining transcription modules using large-scale gene expression data. *Bioinformatics*, pages 1993–2003.

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.