Transcription Module Trees from Gene Expression Data

Gábor Csárdi

October 6, 2009

Contents

1 Introduction

The Iterative Signature Algorithm (ISA) finds transcription modules in gene expression data. A module is a block of the reordered gene expression matrix. The ISA has two threshold parameters: the gene (or feature) threshold and the conditions (or sample) threshold. These essentially specify the stringency of the biclustering. If the gene threshold is high, then modules will have less genes, only the one that are consistently far above or below the mean expression level will be included. The same is true for the sample threshold and the samples in the modules.
It is possible to run the ISA using different threshold parameters and define a hierarchical organization of the found modules. We call such a description a sweep graph or sweep tree. The vertices (or nodes) of the sweep graph are the modules and there is a directed edge from module A to module B if the ISA converges to module B, when started from module A, and the thresholds that were used to find module B are used.

A sweep tree can be generated by keeping one threshold parameter fixed, and vary the other one. We first find modules at a stringent threshold value and then check where these converge at less stringent thresholds.
This document shows how it is possible to create a sweep tree with the `eisa` package and the tree can be visualized.

2 Running the ISA

We will use the acute lymphoblastic leukemia data set from the `ALL` package.

```
> library(ALL)
> library(hgu95av2.db)
> library(genefilter)
> library(eisa)
> library(GO.db)
> data(ALL)

> varLimit <- 0.5
> kLimit <- 4
> ALimit <- 5
```

First we filter the genes and keep only the ones that have an expression level of at least 5 in at least 4 samples; and that have a variance of at least 0.5 across the samples.

```
> flist <- filterfun(function(x) var(x) > varLimit,
     kOverA(kLimit, ALimit))
> ALL.filt <- ALL[genefilter(ALL, flist), ]
```

We remove the T-cell samples as well, and use only the B-cell leukemia samples.

```
> ALL.filt2 <- ALL.filt[, grepl("^B", ALL.filt$BT)]
```

Next, we run the ISA, with keeping the condition threshold fixed, but varying the gene threshold.

```
> set.seed(1)
> modules <- ISA(ALL.filt2, flist = NA, thr.gene = seq(2,
     4, by = 0.5), thr.cond = 1)
```

3 Generating the module tree

The `ISASweep()` function can be used to create a sweep tree from a set of ISA modules. This essentially means finding the edges between the modules and occasionally involves finding new modules as well.

```
> modules2 <- ISASweep(ALL.filt2, modules)
```

`ISASweep()` returns an `ISAModules` object, but adds some new seed data that defines the edges of the sweep graph, plus possibly some new modules:

```
> modules
```

```
An ISAModules instance.
  Number of modules: 36
  Number of features: 1132
  Number of samples: 95
  Gene threshold(s): 4, 3.5, 3, 2.5, 2
  Conditions threshold(s): 1
```

```
> modules2
```

```
An ISAModules instance.
  Number of modules: 43
  Number of features: 1132
  Number of samples: 95
  Gene threshold(s): 4, 3.5, 3, 2.5, 2
  Conditions threshold(s): 1
```

```
> colnames(seedData(modules))
```

```
[1] "iterations"  "oscillation" "thr.row"     "thr.col"
[5] "freq"        "rob"         "rob.limit"
```

```
> colnames(seedData(modules2))
```

```
[1] "iterations"  "oscillation" "thr.row"     "thr.col"
[5] "freq"        "rob"         "rob.limit"   "father"
[9] "level"
```

The `father` column contains pointers to the parent vertices in the graph and the `level` column gives levels defined based on the gene thresholds, higher levels indicate less stringent thresholds.
This data can be converted into a graph and then plotted using the `igraph` package. The `ISASweepGraph()` function creates the graph.

```
> G <- ISASweepGraph(modules2)
```

G has a lot of attributes:

```
> list.graph.attributes(G)
```

```
[1] "layout" "width"  "height"
```

```
> list.vertex.attributes(G)
```

```
[1] "thr"        "id"         "color"      "shape"
[5] "size"       "size2"      "label"      "noFeatures"
[9] "noSamples"
```

The `layout` graph attribute contains a two-column matrix with carefully calculated coordinates for plotting the graph. The `width` and `height` graph attributes contain the suggested size of the plot in inches.

```
> G$width
```

```
[1] 10
```

```
> G$height
```

```
[1] 10.6
```

The `id` vertex attribute contains the ids of the modules. The other vertex attributes essentially define the look of the graph, when it is plotted.
See the documentation of the `igraph` package for more about igraph graph objects.

4 Plotting the sweep tree

4.1 Simple plots

The `ISASweepGraphPlot()` function can be used to plot the graph returned by `ISASweepGraph()`. We will use the suggested size for the plot, and leave all plotting options at their default values. The result is in Fig. 1.

```
> X11(width = G$width, height = G$height)
```

```
> ISASweepGraphPlot(G)
```

4.2 Customized plots

Let us show a little example on how to customize the module tree plot. We will add some annotation to the module, based on Gene Ontology enrichment calculations. Let us perform the hypergeometric test for the enrichment calculation. The p-values are automatically corrected using the Benjamini-Hochberg method.

```
> GO <- ISAGO(modules2)
```

We will color the vertices according to their enrichment in the Biological Process GO ontology. We also create some labels for them, the abbreviated GO terms will be used for this.

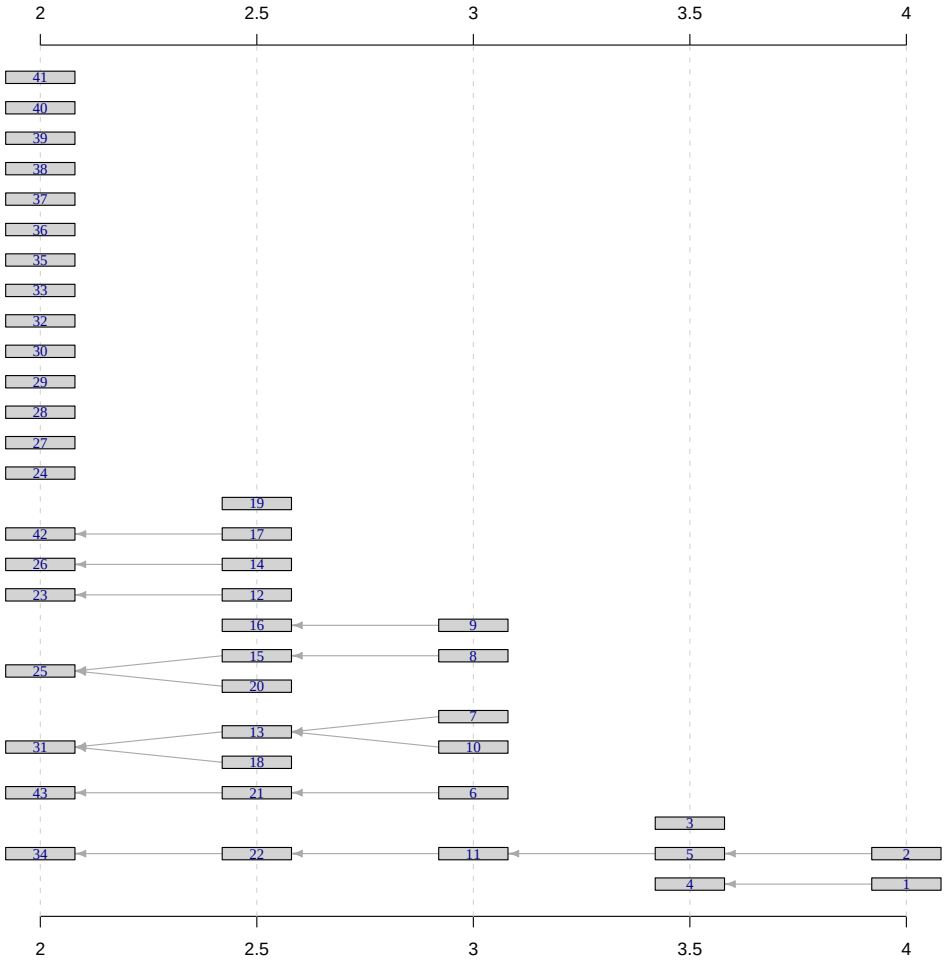

Figure 1: An ISA module tree. Each rectangle symbolizes a transcription module, modules in the same column were found at the same gene threshold. An arrow between two modules indicates that one is the "generalization" of the other, in the sense that ISA converges to the more general (=bigger) module at the milder threshold.

```
> p.bp <- sapply(summary(GO$BP), function(x) as.integer(-log10(x$Pvalue[1])))
> c.bp <- d.bp <- sapply(sigCategories(GO$BP), function(x) x[1])
> d.bp[!is.na(c.bp)] <- sapply(mget(na.omit(c.bp),
    GOTERM), Term)
> d.bp <- abbreviate(d.bp, 6)
> colbar <- hcl(h = 260, c = 35, l = seq(30, 100,
    length = 20))
> colbar <- c("#FFFFFF", rev(colbar))
> col <- colbar[p.bp]

> X11(width = G$width, height = G$height)
```

We are ready to plot the sweep tree now. We put the abbreviated GO terms to the top left corner, and the size of the module (i.e. number of genes and number of samples) in the top right corner. We also increase the height of the vertices a bit.

```
> ISASweepGraphPlot(G, vertex.color = col, vertex.size2 = 50,
    vertex.label.topleft = d.bp, vertex.label.topright = paste(V(G)$noFeatures,
        sep = ",", V(G)$noSamples))
```

Finally, we add a key for the abbreviated names at the top right corner. The result is in Fig. 2.

```
> key <- na.omit(d.bp)[unique(names(na.omit(d.bp)))]
> key2 <- paste(key, sep = ": ", names(key))
> legend("topright", key2)
```

5 More information

For more information about the ISA, please see the references below. The ISA homepage at `http://www.unil.ch/cbg/homepage/software.html` has example data sets, and all ISA related tutorials and papers.

6 Session information

The version number of R and packages loaded for generating this vignette were:

- R version 2.9.2 (2009-08-24), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8;LC_NUMERIC=C;LC_TIME=en_US.UTF-8;LC_COLLATE=en_US.UTF-8;LC_MONETARY=C;LC_MESSAGES=en_US.UTF-8;LC_PAPER=en_US.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_TELEPHONE=C;LC_MEASUREMENT=en_US.UTF-8;LC_IDENTIFICATION=C`

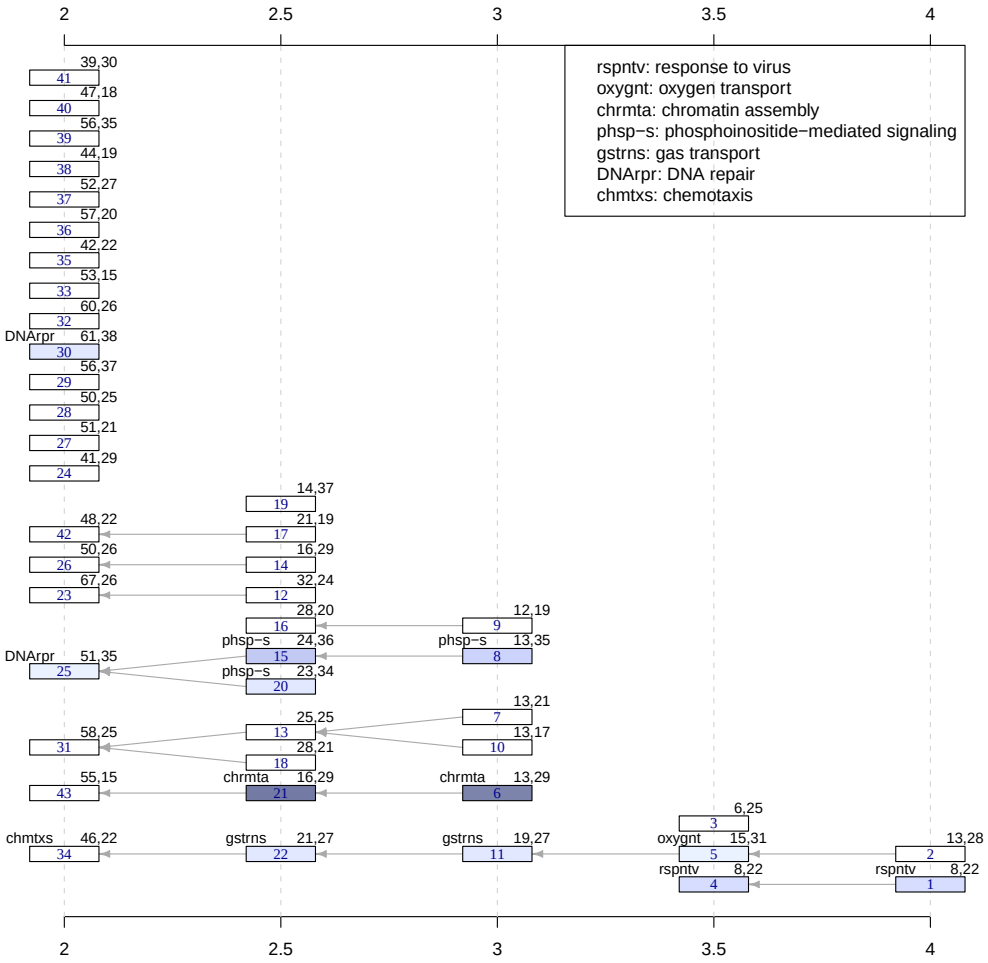

Figure 2: The same module tree, but with some annotation added. The modules are colored according to their Gene Ontology Biological Process enrichment p-values, darker colors correspond to more significant enrichment. The enriched GO terms and the sizes of the modules were also added.

- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: ALL 1.4.4, AnnotationDbi 1.6.0, Biobase 2.4.1, Category 2.10.0, DBI 0.2-4, eisa 0.2, genefilter 1.24.2, GO.db 2.2.5, hgu95av2.db 2.2.12, igraph 0.5.2-2, isa2 0.1, RSQLite 0.7-1
- Loaded via a namespace (and not attached): annotate 1.22.0, biclust 0.8.1, colorspace 1.0-1, graph 1.22.2, grid 2.9.2, GSEABase 1.6.0, MASS 7.2-48, RBGL 1.20.0, splines 2.9.2, survival 2.35-4, tools 2.9.2, vcd 1.2-4, XML 2.6-0, xtable 1.5-5

References

[Bergmann et al., 2003] 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.

[Csárdi, 2009a] Csárdi, G. (2009a). eisa: The iterative signature algorithm for gene expression data. R package version 0.2.

[Csárdi, 2009b] Csárdi, G. (2009b). isa2: The iterative signature algorithm. R package version 0.2.

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

[Ihmels et al., 2002] Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y., and Barkai, N. (2002). Revealing modular organization in the yeast transcriptional network. *Nat Genet*, pages 370–377.

[Kaiser et al., 2009] Kaiser, S., Santamaria, R., Theron, R., Quintales, L., and Leisch, F. (2009). biclust: Bicluster algorithms. R package version 0.7.2.

[Luscher, 2009] Luscher, A. (2009). Expressionview: Visualize overlapping biclusters. R package version 0.2.

[Madeira and Oliveira, 2004] Madeira, S. and Oliveira, A. (2004). Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1:24–45.

[Turner et al., 2003] Turner, H., Bailey, T., and Krzanowski, W. (2003). Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics and Data Analysis*, 48:235–254.