The `eisa` and `biclust` packages

Gábor Csárdi

October 6, 2009

Contents

1 Introduction **1**

2 From `Biclust` to `ISAModules` **2**

3 From `ISAModules` to `Biclust` **7**

4 More information **10**

5 Session information **10**

1 Introduction

Biclustering is technique that simultaneously clusters the rows and columns of a matrix [Madeira and Oliveira, 2004]. In other words, the problem is finding blocks in the reordered input matrix that exhibit correlated behavior, both across the rows and columns of the block.
Biclustering is used increasingly in the analysis of gene expression data sets, because it reduces the complexity of the data: instead of tens of thousands of individual genes, one can focus on a handful of biclusters, in which the genes behave similarly.
The Iterative Signature Algorithm (ISA) [Bergmann et al., 2003] is a biclustering method, that can efficiently find potentially overlapping biclusters (modules, according to the ISA terminology) in a matrix. The ISA is implemented in the `eisa` package [Csárdi, 2009] it uses standard BioConductor classes and includes a number of visualization tools as well.
The `biclust` R package [Kaiser et al., 2009] is a general biclustering package, it contains several biclustering methods, and these can be invoked with a common interface. It provides a set of visualization tools for the results.
In this short document, we show examples on how to use the visualization tools of `eisa` for the biclusters found with `biclust`, and vice-versa.

2 From Biclust to ISAModules

For all examples in this document, we will use the acute lymphoblastic leukemia data set, that is included in the standard BioConductor ALL package. Let's load this data set and the required packages first.

```
> library(biclust)
> library(eisa)
> library(ALL)
> data(ALL)
```

Next, we select a subset of the genes in the data set. We do this to speed up the computation for our simple examples. We select the genes that are annotated to involved in immune system processes, according to the Gene Ontology database.

```
> library(GO.db)
> library(hgu95av2.db)
> gotable <- toTable(GOTERM)
> myterms <- unique(gotable$go_id[gotable$Term %in%
      c("immune system process")])
> myprobes <- unique(unlist(mget(myterms, hgu95av2GO2ALLPROBES)))
> ALL.filtered <- ALL[myprobes, ]
```

We have kept only 970 probes:

```
> nrow(ALL.filtered)
```

```
Features
     970
```

For consistent results, we set the random seed.

```
> set.seed(3840)
```

Next, we apply the Plaid Model Biclustering method [Turner et al., 2003] to the reduced data set.

```
> Bc <- biclust(exprs(ALL.filtered), BCPlaid(),
      fit.model = ~m + a + b, verbose = FALSE)
```

```
Layer Rows Cols   Df          SS      MS Convergence
    0  970  128 1097 4517335.22 4117.90          NA
    1    7   29   35    1505.58   43.02           1
    2   30   28   57    5193.44   91.11           1
    3   11   15   25     214.82    8.59           1

Layer Rows Released Cols Released
    0            NA            NA
    1           109             4
    2            36             6
    3           240            10
```

The method finds 3 biclusters, and returns a `Biclust` object:

```
> class(Bc)

[1] "Biclust"
attr(,"package")
[1] "biclust"

> Bc

An object of class Biclust

call:
        biclust(x = exprs(ALL.filtered), method = BCPlaid(),
            fit.model = ~m + a + b, verbose = FALSE)

Number of Clusters found:  3

First  3  Cluster sizes:
                   BC 1 BC 2 BC 3
Number of Rows:    " 7" "30" "11"
Number of Columns: "29" "28" "15"
```

Now we will convert the `Biclust` object to an `ISAModules` object, that is used in the `eisa` package. To help some `eisa` functions, we add the name of the annotation package to the parameters stored in the `Biclust` object, this is always adwised. The procedure makes use of the probe and sample names that are kept and stored in the `Biclust` object, this information will be used later, e.g. for the enrichment analysis. The conversion itself can be performed with the usual `as()` function.

```
> Bc@Parameters$annotation <- annotation(ALL.filtered)
> modules <- as(Bc, "ISAModules")
> modules

An ISAModules instance.
  Number of modules: 3
  Number of features: 970
  Number of samples: 128
  Gene threshold(s):
  Conditions threshold(s):
```

Now we are able apply the usual `ISAModules` methods to the biclusters. See more about these functions in the documentation of the `eisa` package.
Doing some enrichment analysis is easy:

```
> library(KEGG.db)
> KEGG <- ISAKEGG(modules)
> sigCategories(KEGG)[[2]]
```

```
[1] "04612" "05320" "05332" "04940" "05330" "05310" "04514"
[8] "05322" "04662"
```

```
> unlist(mget(sigCategories(KEGG)[[2]], KEGGPATHID2NAME))
```

```
                                04612
"Antigen processing and presentation"
                                05320
         "Autoimmune thyroid disease"
                                05332
          "Graft-versus-host disease"
                                04940
           "Type I diabetes mellitus"
                                05330
                "Allograft rejection"
                                05310
                              "Asthma"
                                04514
     "Cell adhesion molecules (CAMs)"
                                05322
       "Systemic lupus erythematosus"
                                04662
  "B cell receptor signaling pathway"
```

The `ISA2heatmap()` function creates a heatmap for a module. Let us annotate the heatmap with the leukemia sample type, white means B-cell, black means T-cell leukemia. See Fig. 1.

```
> col <- ifelse(grepl("^B", ALL.filtered$BT), "white",
     "black")
> modcol <- col[getSamples(modules, 2)[[1]]]
> ISA2heatmap(modules, 2, ALL.filtered, ColSideColors = modcol)
```

It turns out, that all samples in the second bicluster belong to patients with T-cell leukemia.
Profile plots visualize the mean expression levels, both for the genes/samples in the module and in the background (i.e. the background means all genes and samples *not* in the module). See Fig. 2.

```
> profilePlot(modules, 2, ALL, plot = "both")
```

The `gograph()` and `gographPlot()` functions create a plot of the part of the Gene Ontology tree that contains the enriched categories. See Fig. 3.

```
> library(GO.db)
> GO <- ISAGO(modules)
> gog <- gograph(summary(GO$CC)[[2]])
> summary(gog)
> gographPlot(gog)
```

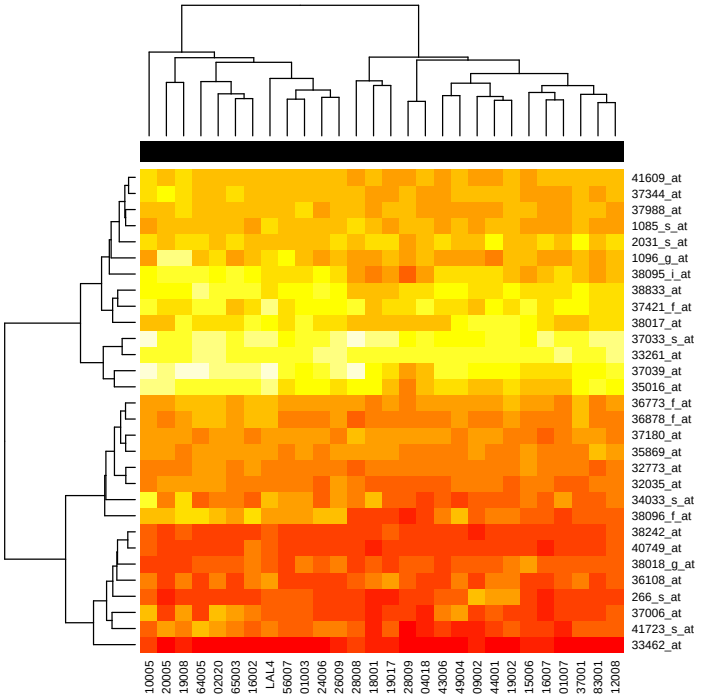

Figure 1: Heatmap of the second module, found with the Plaid Model biclustering algorithm. The black squares denote the T-cell samples; all samples in the module belong to T-cell samples.

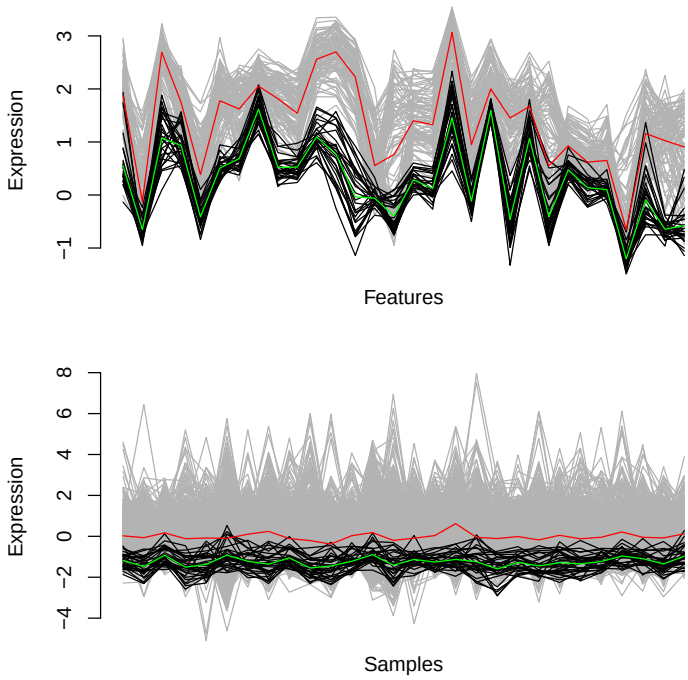

Figure 2: Profile plot for the second module. The red lines show the average exression of the samples/genes in the module. The green lines show the same for the samples/genes not in the module.

```
Vertices: 17
Edges: 16
Directed: TRUE
Graph attributes: width, height, layout.
Vertex attributes: color, name, plabel, label, desc, abbrv, definition, size, size2, shape,
Edge attributes: type, color, arrow.size.
```

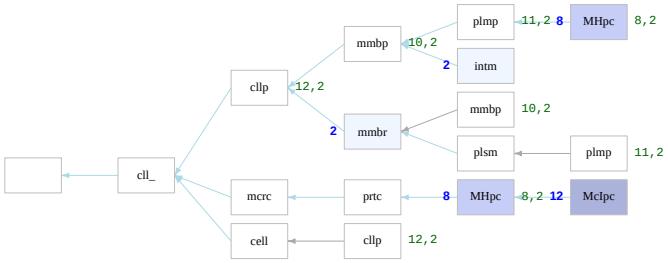

Figure 3: Part of the Gene Ontology tree, Cellular Components ontology. The plot includes all terms with significant enrichment for the second module, and their parent terms, up to the most general term.

The `ISAHTML()` function creates a HTML overview of all modules.

```
> CHR <- ISACHR(modules)
> htmldir <- tempdir()
> ISAHTML(eset = ALL.filtered, modules = modules,
     target.dir = htmldir, GO = GO, KEGG = KEGG,
     CHR = CHR, condPlot = FALSE)
> if (interactive()) {
     browseURL(URLencode(paste("file://", htmldir,
         "/index.html", sep = "")))
 }
```

The `ISAmnplot()` funtion plots group means of expression levels againts each other, for all genes in the module. Here we plot the mean expression of the B-cell samples against the T-cell samples, for the second module. See Fig. 4.

```
> group <- ifelse(grepl("^B", ALL.filtered$BT),
     "B-cell", "T-cell")
> ISAmnplot(modules, 2, ALL.filtered, norm = "raw",
     group = group)
```

3 From `ISAModules` to `Biclust`

It is also possible to convert an `ISAModules` object to a `Biclust` object, but this involves some information loss. The reason for this is, that ISA biclusters

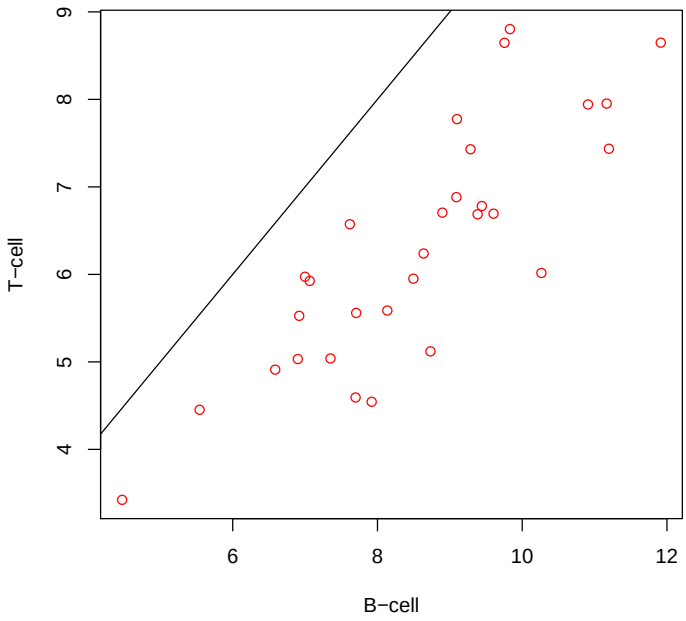

Figure 4: Group means against each other, for B-cell and T-cell samples, for all genes in the second bicluster.

are not binary, but the genes and the samples both have scores between minus one and one; whereas `Biclust` biclusters are required to be binary.
We make use of the small sample set of modules that is included in the `eisa` package. These were generated for the ALL data set.

```
> data(ALLModules)
> ALLModules

An ISAModules instance.
  Number of modules: 82
  Number of features: 3522
  Number of samples: 128
  Gene threshold(s): 4, 3.5, 3, 2.5, 2
  Conditions threshold(s): 3, 2.5, 2, 1.5, 1
```

The conversion from `ISAModules` to `Biclust` can be done the usual way:

```
> BcMods <- as(ALLModules, "Biclust")
> BcMods

An object of class Biclust

call:
        NULL

Number of Clusters found:  82

First  5  Cluster sizes:
                  BC 1 BC 2 BC 3 BC 4 BC 5
Number of Rows:    " 7" " 6" " 2" " 7" "14"
Number of Columns: " 3" " 5" " 5" " 6" " 2"
```

The usual methods of the `Biclust` class can be applied to `BcMods` now. E.g. we can calculate the coherence of the biclusters:

```
> data <- exprs(ALL[featureNames(ALLModules), ])
> constantVariance(data, BcMods, 1)

[1] 2

> additiveVariance(data, BcMods, 1)

[1] 1.4

> multiplicativeVariance(data, BcMods, 1)

[1] 0.14

> signVariance(data, BcMods, 1)
```

```
[1] 0.92
```

As another example, we calculate these coherence measures for all modules and compare them to the ISA robustness measure.

```
> cV <- sapply(1:BcMods@Number, function(x) constantVariance(data,
     BcMods, x))
> aV <- sapply(1:BcMods@Number, function(x) additiveVariance(data,
     BcMods, x))
> mV <- sapply(1:BcMods@Number, function(x) multiplicativeVariance(data,
     BcMods, x))
> sV <- sapply(1:BcMods@Number, function(x) signVariance(data,
     BcMods, x))
> rob <- ISARobustness(ALL, ALLModules)
```

Let's create a pairs-plot to visualize the relationship of these measures for our data set, the result is in Fig. 5.

```
> pairs(cbind(cV, aV, mV, sV, rob))
```

4 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.

5 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`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: ALL 1.4.4, AnnotationDbi 1.6.0, biclust 0.8.1, Biobase 2.4.1, Cairo 1.4-4, Category 2.10.0, colorspace 1.0-1, 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, KEGG.db 2.2.5, MASS 7.2-48, org.Hs.eg.db 2.2.11, RSQLite 0.7-1, vcd 1.2-4, xtable 1.5-5

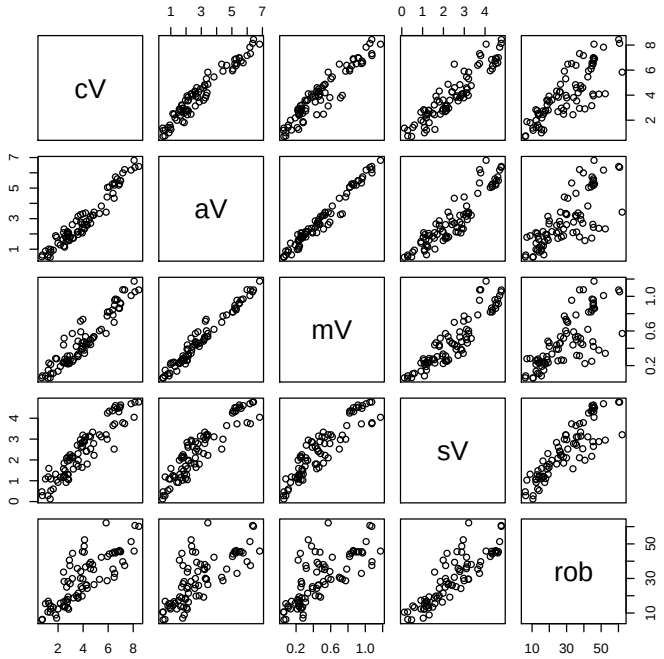

Figure 5: Relationship of the various bicluster coherence measueres and the ISA robustness measure. They show high correlation.

- Loaded via a namespace (and not attached): annotate 1.22.0, graph 1.22.2, GSEABase 1.6.0, RBGL 1.20.0, splines 2.9.2, survival 2.35-4, tools 2.9.2, XML 2.6-0

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

[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.

[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.