Tissue selective expression with the Iterative Signature Algorithm

Gábor Csárdi

September 30, 2013

Contents

1 Introduction

The Iterative Signature Algorithm (ISA) is a biclustering method. It finds genes that are up- and/or downregulated consistently in a subset of samples, in gene expression data. (Although, it can be also applied to other tabular data sets.) Please see [Bergmann et al., 2003] for the details about the ISA, and also the other tutorials at the ISA homepage, `http://www.unil.ch/cbg/ISA`.

In this tutorial we show an example application of the ISA, for which we use a publicly available data set, a panel of murine tissues. Basic understanding of R and the ISA is a prerequisite for this tutorial, please see the introductory ISA tutorials at the ISA homepage first.

We use a number of R packages for this tutorial, all of them are available from the standard BioConductor or CRAN repositories. We will use the `GEOquery` package [Davis and Meltzer, 2007] to download the raw data files for the tissue-panel experiment. The `eisa` package contains the implementation of the ISA. The `affy` package [Gautier et al., 2004] is used to read in

the CEL files and calculate the MAS5 Present/Absent calls; the `gcrma` package contains the implementation of the GCRMA gene expression normalization method [Wu et al., 2003]. `GO.db` contains the Gene Ontology database [The Gene Ontology Consortium, 2000], `KEGG.db` the KEGG Pathway database [Kanehisa and Goto, 2000]. The `xtable` package is used to create a nicely formatted table.

Let's load all these packages first.

```
> library(GEOquery)
> library(eisa)
> library(affy)
> library(gcrma)
> library(GO.db)
> library(KEGG.db)
> library(xtable)
```

2 Loading the data set

First we define the data set that we want to use. This is a murine tissue panel for 22 tissues, 3-5 samples each, 70 samples altogether. The different tissues were dissected from 10-12 week old C57Bl6 mice for RNA extraction and hybridization on Affymetrix microarrays. This is the ID of the experiment in the Gene Expression Omnibus (GEO) [Barrett et al., 2009]:

```
> GEO <- "GSE9954"
```

To spare time and bandwidth, we download the data only once and store it in the local disk, this facilitates running this tutorial more than once. The `getGEOSuppFile()` function from the `GEOquery` package creates a directory in the current working directory, names it according to the GEO id of the experiment and stores the raw files there. Then, the `untar()` function unpacks the downloaded archive. If this directory already exists, then we have nothing to download or unpack.

```
> if (!file.exists(GEO)) {
   rawfiles <- getGEOSuppFiles(GEO)
   tarfile <- grep("\\.tar$", rownames(rawfiles), value=TRUE)
   untar(tarfile, exdir=GEO)
 }
```

Next, we read in the downloaded and unpacked CEL files and store the data in an `AffyBatch` object. Because this is a quite lengthty process, we perform it only once and store the results in a file, in the same directory as the raw CEL files. We don't have to do anything if this file already exists.

```
> ABfile <- paste(GEO, sep="/", "AB.Rdata")
> if (!file.exists(ABfile)) {
```

```
   celfiles <- list.files(GEO, pattern="\\.CEL.gz", full.names=TRUE)
   AB <- ReadAffy(filenames=celfiles)
   save(AB, file=ABfile)
 }
```

To get rid of the probesets that do not match a gene that is expressed in any of the tissues, we calculate the MAS5 Present/Absent calls. These are based on the perfect match and mismatch probes on the array and provide a way to access the probes that are present in a tissue. We save the results in a file. If the file already exists, then no calculation is needed, and we just load it.

```
> PAfile <- paste(GEO, sep="/", "PA.Rdata")
> if (!file.exists(PAfile)) {
   load(ABfile)
   PA <- mas5calls(AB)
   save(PA, file=PAfile)
 } else {
   load(PAfile)
 }
```

GCRMA normalization [Wu et al., 2003] is the next step of the analysis. This results the background corrected, log2 transformed expression values. Running GCRMA takes a relatively long time, so we save these results in a file, as well. If the file is already present, we just load it. `EXP` is an `ExpressionSet` object here, it contains all the expression data, and may contain experiment meta-data as well.

```
> EXPfile <- paste(GEO, sep="/", "EXP.Rdata")
> if (!file.exists(EXPfile)) {
   tissueEXP <- gcrma(AB, fast=FALSE)
   save(tissueEXP, file=EXPfile)
 } else {
   load(EXPfile)
 }
```

Next, we do the filtering, based on the previously calculated detection calls. We keep only probesets that were called 'Present' in at least three samples.

```
> keep <- rowSums(exprs(PA) == "P") >= 3
> EXP <- tissueEXP[keep,]
```

Next, we load the annotation package of the array that was used in the experiment. BioConductor provides annotation packages for all the standard (and many non-standard) arrays. These packages include mappings from the array probe ids to Entrez (and other) gene ids, and also mappings to databases. The `annotation()` function gives the name of the annotation package. If you don't have the required annotation package installed, then please install it, the same way as you install normal BioConductor packages.

```
> annotation(EXP)

[1] "mouse4302"

> annpackage <- paste(annotation(EXP), sep=".", "db")
> library(annpackage, character.only=TRUE)
```

Unfortunately, the files downloaded from GEO do not contain any meta data. We, however, need the tissue information for each sample, so we have to download these separately. First we create a function (`annotSample()`) that takes a sample name and downloads its corresponding meta data from GEO. We only use the "sample source" (=tissue) information from the meta data.

We put the tissue labels into the `ExpressionSet` object, and save the new version. If these labels are already present in the `ExpressionSet`, then we don't have to do anything.

```
> if (is.null(EXP$tissue)) {
   annot <- getGEOfile(GEO, amount="quick")
   annotSample <- function(x) {
     x <- sub(".CEL.gz", "", x, fixed=TRUE)
     f <- getGEOfile(x, amount="quick")
     l <- readLines(f)
     s <- grep("Sample_source_name", l, value=TRUE)[1]
     strsplit(s, " = ")[[1]][2]
   }
   tissues <- sapply(sampleNames(EXP), annotSample)
   tissueEXP$tissue <- sub("Mouse ", "", tissues, fixed=TRUE)
   EXP$tissue <- tissueEXP$tissue

   save(tissueEXP, file=EXPfile)
 }
```

3 Running the ISA

We are ready to run the ISA on the data. First we calculate the ISA normalized expression matrix, see the documentation of the `ISANormalize()` function for details. (It is possible to use the un-normalized expression matrix in any of following examples, but pre-calculating it saves some time later.)

We run the ISA with two threshold combinations. The gene threshold is fixed at 3. (Meaning that the genes of a module must be at least 3 standard deviations away from the (weighted) mean expression in the samples of the module.) The condition threshold is set to 1 or 2. We expect to get smaller modules (in terms of samples) for the stricter threshold. As before, we save the modules in a file, to save a couple of minutes.

```
> NEXP <- ISANormalize(EXP)
> modulesfile <- paste(GEO, sep="/", "modules.Rdata")
```

```
> if (!file.exists(modulesfile)) {
   set.seed(123)
   modules <- ISA(EXP, flist=NA, thr.gene=3, thr.cond=c(1,2), no.seeds=1000)
   save(modules, file=modulesfile)
 } else {
   load(modulesfile)
 }
```

Note, that we set the seed of the random number generator here with the `set.seed()` function, to get the same results every time we run the code of this tutorial. This is not needed for regular analysis. Also note, that even if the random seed is the same, you might still get different results (=modules), depending on your R version.

Let's quickly check the modules, found by ISA.

```
> length(modules)

[1] 26

> getNoFeatures(modules)

 [1] 195 253 248 345 366 285 188 331 246 208 370 264 180 350
[15] 212 249 181 172 265 173 118 144 170 143 304 231

> getNoSamples(modules)

 [1]  5  3  3  3  3  3  3  3  3  4  3  3  3  3  3  3  7  3
[19]  3  3 10 10  6 10  6  8
```

We have 26 altogether. The largest one contains 370 genes, the smallest one has 118 genes. As for the samples, most modules contain three samples only, these should correspond to a single tissue, we will check this in a minute. Some modules have more than three samples, these probably contain genes that have high (or low) expression in multiple tissues.

4 Finding tissue selective modules

Tissue selectivity of a gene means that the gene has a higher expression in one tissue, than in other tissues; even if it is expressed in a range of tissues. There are various methods for testing the tissue selectivity of genes, see e.g. [Van Deun et al., 2009] for a method that is well principled and efficient in finding such genes.

The approach we are taking here with ISA is a bit different. We do not ask the question of tissue-selectivity directly, but use an unsupervised method to group genes together that are up- and downregulated together, in some samples. These genes and samples form transcription modules. If it turns out that the samples of a module correspond to a given tissue, then a module of tissue-selective genes was found. It is also very well possible that the

ISA finds modules that are selective for two or three (or four, etc) tissues; the genes of these modules are harder to find via direct testing, one would have to test for all pairs, triples, etc. of tissues to find them.

Condition plots are good for finding tissue selective modules. A condition plot is a barplot of sample scores, potentially also containing (non-zero) scores for the samples that are not included in the module. Let's create a condition plot for a transcription module, module 23.

First, we define some parameters that make the plot look better. We group the samples according to tissues and also assign different colors to the tissues. Please see the documentation of the `condPlot()` function for details about these parameters.

```
> sep <- c(which(!duplicated(NEXP$tissue))[-1]-1, ncol(NEXP))
> names(sep) <- NEXP$tissue[!duplicated(NEXP$tissue)]
> colbar <- rainbow(length(sep))
> col <- colbar[as.factor(NEXP$tissue)]
```

We are ready to create the plot, see the results in Fig. 1.

```
> condPlot(modules, mymod, NEXP, sep=sep, col=col)
```

This module contains samples from 2 tissues: brain, lung.

Now we look for tissue selective modules. In order to consider a module tissue selective, we require that the non-zero sample scores of a tissue have the same signs; i.e. the genes of the module are regulated the same way in all samples of the tissue. We also mark oppositely regulated tissues with a 'd' in parentheses.

```
> tspec <- function(mod) {
   t <- EXP$tissue[getSamples(modules, mod)[[1]]]
   s <- getSampleScores(modules, mod)[[1]]
   t[ s<0 ] <- paste(t[s<0], sep=" ", "(d)")
   unique(t)
 }
> spec <- sapply(seq_len(length(modules)), tspec)
> specc <- sapply(spec, paste, collapse=", ")
> specc

 [1] "pituitary gland"
 [2] "ovary (d)"
 [3] "salivary gland"
 [4] "liver"
 [5] "brain"
 [6] "lung"
 [7] "adrenal gland (d)"
 [8] "eye"
 [9] "seminal vesicle"
[10] "bone marrow"
```

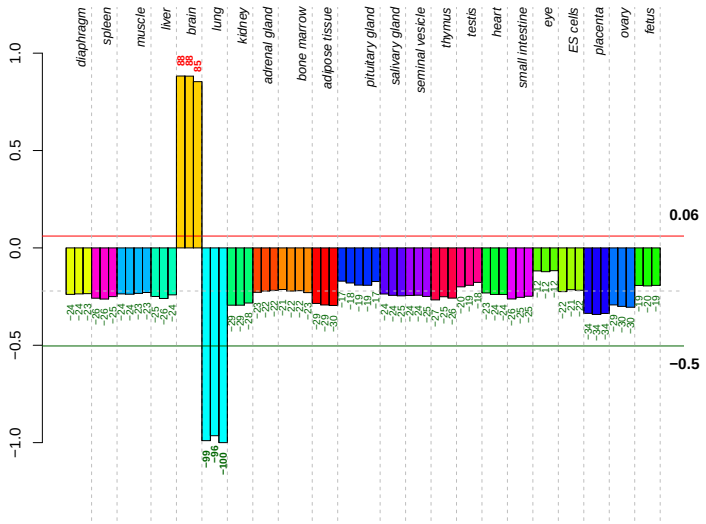

Figure 1: Condition plot for module 23. This module has 170 genes and 6 samples. The samples belong to two tissues: adipose tissue and ES cells. (ES cells were considered as a separate tissue in the experiment.) Out of the 170 genes of the module, 109 are upregulated in the ES cell samples and down-regulated in the adipose tissue samples. The rest of the genes, 61, have opposite regulation.

```
[11] "kidney"
[12] "placenta"
[13] "fetus"
[14] "small intestine"
[15] "thymus (d)"
[16] "spleen"
[17] "diaphragm, muscle"
[18] "ES cells (d)"
[19] "adipose tissue"
[20] "heart"
[21] "diaphragm, muscle, heart"
[22] "spleen, bone marrow, thymus"
[23] "brain, lung (d)"
[24] "diaphragm, muscle, heart"
[25] "lung (d), adipose tissue"
[26] "brain, pituitary gland (d)"
```

19 modules contain a single tissue only, 4 contain exactly two tissues, regulated either the same way, or oppositely. 3 contain three or more tissues.

As another example, we create a plot for four modules that are selective for a single tissue.

```
> to.plot <- which(sapply(spec, length)==1)[1:4]
> layout(rbind(1:2,3:4))
> for (p in to.plot) {
   condPlot(modules, p, NEXP, sep=sep, col=col)
 }
```

5 Enrichment analysis

It is easy to perform enrichment calculations to check whether the tissue selective genes are actually on a common pathway. As usual, we save the results to a file. Note, that the `ISAGO()` and `ISAKEGG()` functions also perform multiple testing correction.

```
> enrichFile <- paste(GEO, sep="/", "enrichment.Rdata")
> if (!file.exists(enrichFile)) {
   GO <- ISAGO(modules)
   KEGG <- ISAKEGG(modules)
   save(GO, KEGG, file=enrichFile)
 } else {
   load(enrichFile)
 }
> GO

$BP
Gene to GO List BP  test for over-representation
```

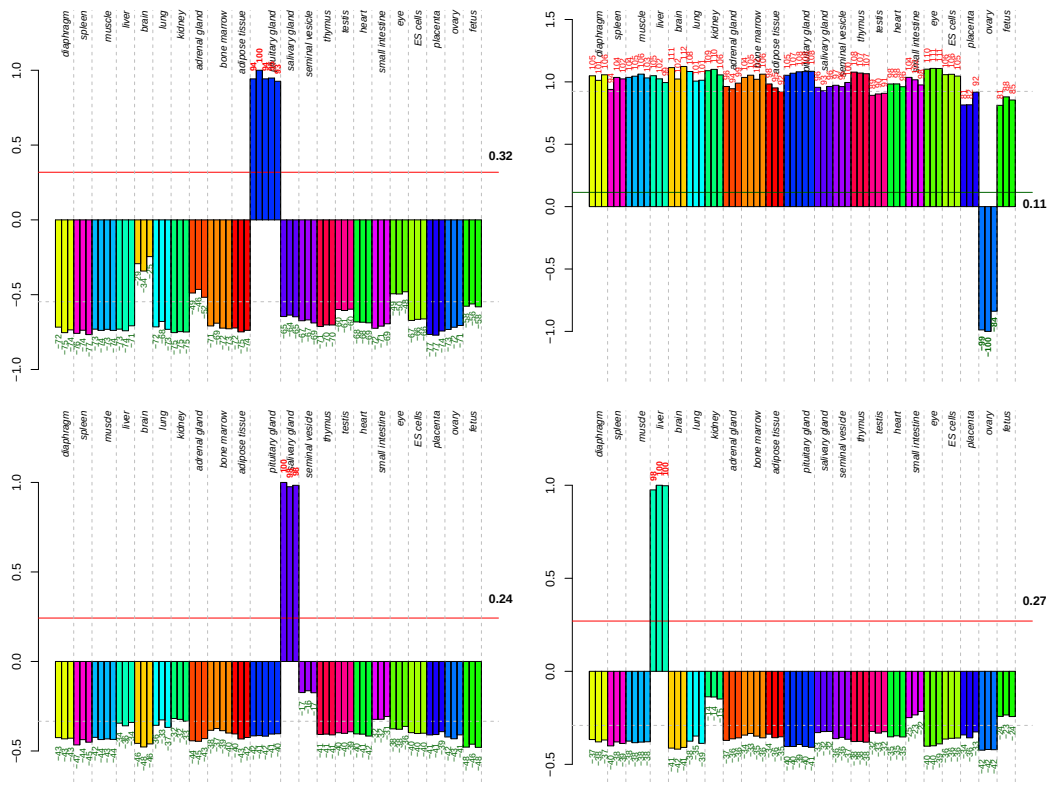

Figure 2: Condition plots for four modules that are selective for a single tissue, modules: 1, 2, 3, 4 (from top to bottom and left to right). They are selective to pituitary gland, ovary, salivary gland, liver, respectively.

```
57904 GO List BP ids tested (0-197 have p < 0.05)
Selected gene set sizes: 108-338
     Gene universe size: 16051
     Annotation package: mouse4302

$CC
Gene to GO List CC  test for over-representation
6144 GO List CC ids tested (0-47 have p < 0.05)
Selected gene set sizes: 111-345
     Gene universe size: 16283
     Annotation package: mouse4302

$MF
Gene to GO List MF  test for over-representation
11735 GO List MF ids tested (0-46 have p < 0.05)
Selected gene set sizes: 108-336
     Gene universe size: 15887
     Annotation package: mouse4302
```

```
> KEGG

Gene to   test for over-representation
2030  ids tested (0-22 have p < 0.05)
Selected gene set sizes: 21-165
     Gene universe size: 4876
     Annotation package: mouse4302
```

Most of the modules are enriched with some Gene Ontology term and/or KEGG pathway. Let's create a table that lists the tissues associated with a module, together with the enriched GO terms and KEGG pathways.

First we define two functions that translate GO and KEGG ids to category and pathway names. These functions also handle `NA` values.

```
> goterm <- function(x) {
   gt <- sapply(mget(na.omit(x), GOTERM), Term)
   res <- character(length(x))
   res[!is.na(x)] <- gt
   res[is.na(x)] <- NA
   res
 }
> keggpath <- function(x) {
   kp <- unlist(mget(na.omit(x), KEGGPATHID2NAME))
   res <- character(length(x))
   res[!is.na(x)] <- kp
   res[is.na(x)] <- NA
   res
 }
```

The `first()` function takes a list of vectors and returns the first element of each vector, we will use this later, to query the most significantly enriched terms.

```
> first <- function(x) if (length(x) >= 1) x[[1]] else NA
```

We are ready to create the list of the most significantly enriched GO terms, for each module. We use the `summary()` function to create a table for the enrichment and take the best p-value for each ontology. We choose the ontology with the best p-value then. Finally, we also query the id of the best enrichment, using the `sigCategories()` function and use the `goterm()` function that we just defined to translate it to a category name.

```
> topgo <- sapply(seq_len(length(modules)), function(x) {
   p <- c(BP=summary(GO$BP, p=2)[[x]][1,]$Pvalue,
          CC=summary(GO$CC, p=2)[[x]][1,]$Pvalue,
          MF=summary(GO$MF, p=2)[[x]][1,]$Pvalue)
   mc <- names(which.min(p))
   paste(goterm(sigCategories(GO[[mc]], p=2)[[x]][1]),
         sep=", p=", format(min(p), digits=3))
 })
```

We do a similar thing for KEGG now. This is simpler, as we don't have multiple ontologies here, all we need is the name and p-value of the most significant enrichment.

```
> keggterms <- keggpath(sapply(sigCategories(KEGG, p=2), first))
> keggp <- sapply(summary(KEGG, p=2), function(x) x$Pvalue[1])
```

Everything is set for creating the table. It has three columns, the name(s) of the tissue(s), the Gene Ontology enrichment and the KEGG enrichment.

```
> eTable <- data.frame(Tissue=specc, GO=topgo,
      KEGG=paste(keggterms, sep=", p=", format(keggp, digits=3)))
```

We use the `xtable` package to create a nice printed version of the table. Because GO category names tend to be long, we typeset the table in landscape mode. In order to make it fit into this document, we only include the first 20 modules.

```
> caption <- paste("The most significantly enriched GO categories",
                   "and KEGG pathways for the (first 20)",
                   "transcription modules.",
                   "Oppositely regulated tissues are marged with `d'.")
> if (nrow(eTable)>20) { eTable2 <- eTable[1:20,] }
> print(xtable(eTable2, caption=caption,
              align=c("r@{\\hspace{1em}}",
                      "p{4.5cm}@{\\hspace{1em}}",
                      "p{7.5cm}@{\\hspace{1em}}", "p{7.5cm}")),
       floating.environment="sidewaystable",
       tabular.environment="tabular")
```

6 Session information

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

- R version 3.0.2 (2013-09-25), `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=en_US.UTF-8`, `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, methods, parallel, stats, utils
- Other packages: affy 1.39.4, AnnotationDbi 1.23.25, Biobase 2.21.7, BiocGenerics 0.7.5, DBI 0.2-7, eisa 1.13.0, gcrma 2.33.1, GEOquery 2.27.2, GO.db 2.10.1, isa2 0.3.3, KEGG.db 2.10.1, mouse4302.db 2.10.1, org.Mm.eg.db 2.10.1, RSQLite 0.11.4, xtable 1.7-1
- Loaded via a namespace (and not attached): affyio 1.29.5, annotate 1.39.0, BiocInstaller 1.11.4, Biostrings 2.29.19, Category 2.27.3, genefilter 1.43.0, graph 1.39.3, grid 3.0.2, GSEABase 1.23.2, IRanges 1.19.38, lattice 0.20-23, Matrix 1.0-14, preprocessCore 1.23.0, RBGL 1.37.2, RCurl 1.95-4.1, splines 3.0.2, stats4 3.0.2, survival 2.37-4, tools 3.0.2, XML 3.98-1.1, XVector 0.1.4, zlibbioc 1.7.0

References

[Barrett et al., 2009] Barrett, T., Troup, D., Wilhite, S., Ledoux, P., Rudnev, D., Evangelista, C., Kim, I., Soboleva, A., Tomashevsky, M., Marshall, K., Phillippy, K., Sherman, P., Muertter, R., and R, E. (2009). NCBI GEO: archive for high-throughput functional genomic data. *Nucleic Acids Research*, 37:D885–90.

[Bergmann et al., 2003] Bergmann, S., Ihmels, J., and Barkai, N. (2003). Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical Review E*, 67:031902.

[Davis and Meltzer, 2007] Davis, S. and Meltzer, P. (2007). GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. *Bioinformatics*, 14:1846–1847.

[Gautier et al., 2004] Gautier, L., Cope, L., Bolstad, B. M., and Irizarry, R. A. (2004). affy—analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics*, 20(3):307–315.

[Kanehisa and Goto, 2000] Kanehisa, M. and Goto, S. (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28:27–30.

[The Gene Ontology Consortium, 2000] The Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

[Van Deun et al., 2009] Van Deun, K., Hoijtink, H., Thorrez, L., Van Lommel, L., Schuit, F., and Van Mechelen, I. (2009). Testing the hypothesis of tissue selectivity: the intersection-union test and a bayesian approach. *Bioinformatics*, 25:2588–2594.

[Wu et al., 2003] Wu, Z., Irizarry, R., Gentleman, R., Murillo, F., and Spencer, F. (2003). A model based background adjustment for oligonucleotide expression arrays. Technical report, John Hopkins University, Department of Biostatistics Working Papers, Baltimore, MD.

	Tissue	GO	KEGG
1	pituitary gland	G-protein coupled receptor signaling pathway, p=3.21e-08	Neuroactive ligand-receptor interaction, p=9.80e-14
2	ovary (d)	reproductive structure development, p=3.63e-10	Steroid hormone biosynthesis, p=7.69e-07
3	salivary gland	extracellular region, p=2.91e-05	Arachidonic acid metabolism, p=1.44e-01
4	liver	extracellular region, p=4.19e-35	Complement and coagulation cascades, p=2.30e-28
5	brain	synapse, p=8.01e-34	Neuroactive ligand-receptor interaction, p=1.87e-19
6	lung	extracellular region, p=6.82e-16	Taste transduction, p=6.69e-03
7	adrenal gland (d)	hormone biosynthetic process, p=1.88e-06	Steroid hormone biosynthesis, p=3.07e-04
8	eye	visual perception, p=1.46e-69	Phototransduction, p=9.03e-21
9	seminal vesicle	sequence-specific DNA binding transcription factor activity, p=0.0724	Glycosphingolipid biosynthesis - globo series, p=2.90e-02
10	bone marrow	immune system process, p=2.64e-22	Hematopoietic cell lineage, p=5.01e-05
11	kidney	secondary active transmembrane transporter activity, p=4.74e-25	Metabolic pathways, p=6.98e-06
12	placenta	hormone activity, p=2.16e-18	Amoebiasis, p=2.45e-02
13	fetus	extracellular region part, p=2.13e-15	Protein digestion and absorption, p=1.36e-06
14	small intestine	intestinal absorption, p=1.01e-09	Fat digestion and absorption, p=1.29e-09
15	thymus (d)	T cell activation, p=8.68e-33	T cell receptor signaling pathway, p=2.06e-16
16	spleen	immune system process, p=4.2e-43	Cytokine-cytokine receptor interaction, p=1.97e-11
17	diaphragm, muscle	myofibril, p=9.41e-39	Hypertrophic cardiomyopathy (HCM), p=1.06e-08
18	ES cells (d)	nucleic acid binding, p=9.42e-10	Ribosome biogenesis in eukaryotes, p=5.47e-02
19	adipose tissue	intrinsic to membrane, p=8.96e-08	PPAR signaling pathway, p=5.83e-04
20	heart	myofibril, p=1.25e-10	Dilated cardiomyopathy, p=2.69e-07

Table 1: The most significantly enriched GO categories and KEGG pathways for the (first 20) transcription modules. Oppositely regulated tissues are marged with 'd'.