DEGraph: differential expression testing for gene networks

Laurent Jacob Pierre Neuvial Sandrine Dudoit

May 3, 2016

Abstract

DEGraph implements recent hypothesis testing methods which directly assess whether a particular gene network is differentially expressed between two conditions. This is to be contrasted with the more classical two-step approaches which first test individual genes, then test gene sets for enrichment in differentially expressed genes. These recent methods take into account the topology of the network to yield more powerful detection procedures. In practice, *DEGraph* makes it very simple to test all KEGG pathways for differential expression on any gene expression data set and provides tools to visualize the results.

1 Introduction

Measuring gene expressions to study a biological phenomenon or build prognosis tools is now common practice. Technologies like DNA microarrays or RNA-Seq allow to systematically measure the expression of thousands of genes in a sample. Statistical univariate statistical procedures like the t-test are then classically applied to detect differentially expressed genes. However when analyzing this type of data, one is very often interested in the "systems biology" approach of detecting pre-defined sets of genes that are known to work together and are significantly differentially expressed between the studied conditions. Two paradigms have been used so far :

- Most approaches are two-step. They start by assessing which genes are differentially expressed, then test gene sets for enrichment in differentially expressed genes.
- Some approaches directly use multivariate statistics on gene sets as vectors of genes to determine whether the multivariate expression of a gene set varies between groups of samples.

Limitations of the former approach have been widely studied (**?**) : that the former family of approaches can lead to incorrect interpretations, as the sampling units for the tests in the second step become the genes (as opposed to the patients) and these are expected to have strongly correlated expression measures. This suggests that direct multivariate testing of

gene set differential expression is more appropriate than posterior aggregation of individual gene-level tests. On the other hand, while multivariate statistics are known to perform well in small dimensions, they lose power very quickly with increasing dimension (**?**).

At the same time, an increasing number of regulation networks are becoming available, specifying, for example, which genes activate or inhibit the expression of which other genes. The method implemented in *DEGraph* intends to use these networks to build spaces of lower dimension, yet retaining most of the expression shift of gene sets. This makes the multivariate testing amenable and provably more powerful under (partly) coherent expression shift assumption.

2 Software features

DEGraph offers the following functionalities:

Multivariate testing *DEGraph* proposes functions to test whether a set of genes organised in a particular network are differentially expressed between two conditions (according to a particular data set of samples).

Interfacing with KEGGgraph The package also provides functions to easily load a set of KEGG networks as `KEGGgraph` objects and systematically test each of their connected components for differential expression with various statistics.

Visualization *DEGraph* provides functions to visualize tested KEGG graphs with nodes colored according to a quantitative variable, typically the individual t-statistics or mean difference of expression between the two conditions for each gene.

3 Case studies

We now show on a simple example how *DEGraph* can be used to assess differential expression of some KEGG pathways using several test statistics, compare and plot the results.

3.1 Loading the library and the data

We load the *DEGraph* package by typing or pasting the following codes in R command line:

```
> library(DEGraph)
```

We then load some more libraries :

```
> library("R.utils")
> ##library(graph)
> ##library(rrcov) ## for 'T2.test'
```

```
> library(corpcor)
> library(KEGGgraph)
> library(Rgraphviz)
> ##library(RBGL)
> library(fields) # For image.plot called in plotValuedGraph
> library(lattice)
> library(marray)
> verbose <- TRUE
```

In this example, the expression and annotation data as well as the list of KEGG networks has been pre-stored in an `.RData` file to avoid lengthy downloading and formatting. For examples on how to build these variables, see the `Loi2008` demo in the package.

```
> data("Loi2008_DEGraphVignette", package="DEGraph")
> classData <- classLoi2008
> exprData <- exprLoi2008
> annData <- annLoi2008
> grList <- grListKEGG
```

3.2 Hypothesis testing

We now run several tests on the expression data restricted to the two selected KEGG networks stored in `grListKEGG`. We start with individual t-tests on all the genes, which will later be used for visualization.

```
> ## Individual t-test p-values
> X1 <- t(exprData[, classData==0])
> X2 <- t(exprData[, classData==1])
> ttpv <- c()
> tts <- c()
> for(i in 1:ncol(X1)) {
+   tt <- t.test(X1[,i],X2[,i])
+   ttpv[i]=unlist(tt$p.value)
+   tts[i]=unlist(tt$statistic)
+ }
> names(ttpv) <- names(tts) <- rownames(exprData)
```

We then run two different multivariate tests on each network:

- Hotelling T^2 test, which generalizes the t-test to multivariate data, and does not make use of the network information.
- Hotelling T^2 test in a space of lower dimension built from the network structure.

```
> prop <- 0.2
> ## Multivariate tests
> resList <- NULL
> for (ii in seq(along=grList)) {
+   gr <- grList[[ii]]
+   res <- testOneGraph(gr, exprData, classData, verbose=verbose, prop=prop)
+   resList <- c(resList, list(res))
+ }
> resNames <- names(grList)
> pLabels <- sapply(grList, attr, "label")
> ## get rid of NULL results (no connected component of size > 1)
> isNULL <- sapply(resList, is.null)
> if (sum(isNULL)) {
+   grList[isNULL]
+   resList <- resList[!isNULL]
+   resNames <- names(grList)[!isNULL]
+   pLabels <- pLabels[!isNULL]
+ }
> resL <- sapply(resList, length)
> graphNames <- rep(resNames, times=resL)
> pathwayNames <- rep(pLabels, times=resL)
> graphList <- NULL
> for (res in resList) {
+   grl <- lapply(res, FUN=function(x) {
+     x$graph
+   })
+   graphList <- c(graphList, as.list(grl))
+ }
> ndims <- NULL
> for (res in resList) {
+   ndim <- sapply(res, FUN=function(x) {
+     x$k
+   })
+   ndims <- c(ndims, ndim)
+ }
> pKEGG <- NULL
> for (res in resList) {
+   pp <- sapply(res, FUN=function(x) {
+     x$p.value
+   })
+   pKEGG <- cbind(pKEGG, pp)
```

```
+ }
> colnames(pKEGG) <- graphNames
> rn <- rownames(pKEGG)
> rownames(pKEGG)[grep("Fourier", rn)] <- paste("T2 (", round(100*prop), "% Fourier compone
> if (exists("maPalette", mode="function")) {
+   pal <- maPalette(low="red", high="green", mid="black", k=100)
+ } else {
+   pal <- heat.colors(100)
+ }
> shift <- tts # Plot t-statistics
> names(shift) <- translateGeneID2KEGGID(names(tts))
> fSignif <- which(pKEGG[2,] < 0.05)
> fSignif <- fSignif[order(pKEGG[2,fSignif])]
```

Finally we plot two of the tested networks along with the p-value for differential expression for each of the two multivariate statistics. Node colors correspond to the gene t-statistic :

```
> gIdx <- fSignif[1]
> gr <- graphList[[gIdx]]
> mm <- match(translateKEGGID2GeneID(nodes(gr)), rownames(annData))
> dn <- annData[mm, "NCBI.gene.symbol"]
> res <- plotValuedGraph(gr, values=shift, nodeLabels=dn, qMax=0.95, colorPalette=pal, heig
> stext(side=3, pos=0, pathwayNames[gIdx])
> ps <- signif(pKEGG[, gIdx],2)
> txt1 <- paste("p(T2)=", ps[1], sep="")
> txt2 <- paste("p(T2F[", ndims[gIdx], "])=", ps[2], sep="")
> txt <- paste(txt1, txt2, sep="\n")
> stext(side=3, pos=1, txt)
> image.plot(legend.only=TRUE, zlim=range(res$breaks), col=pal, legend.shrink=0.3, legend.w

> gIdx <- fSignif[5]
> gr <- graphList[[gIdx]]
> mm <- match(translateKEGGID2GeneID(nodes(gr)), rownames(annData))
> dn <- annData[mm, "NCBI.gene.symbol"]
> res <- plotValuedGraph(gr, values=shift, nodeLabels=dn, qMax=0.95, colorPalette=pal, heig
> stext(side=3, pos=0, pathwayNames[gIdx])
> ps <- signif(pKEGG[, gIdx],2)
> txt1 <- paste("p(T2)=", ps[1], sep="")
> txt2 <- paste("p(T2F[", ndims[gIdx], "])=", ps[2], sep="")
> txt <- paste(txt1, txt2, sep="\n")
```

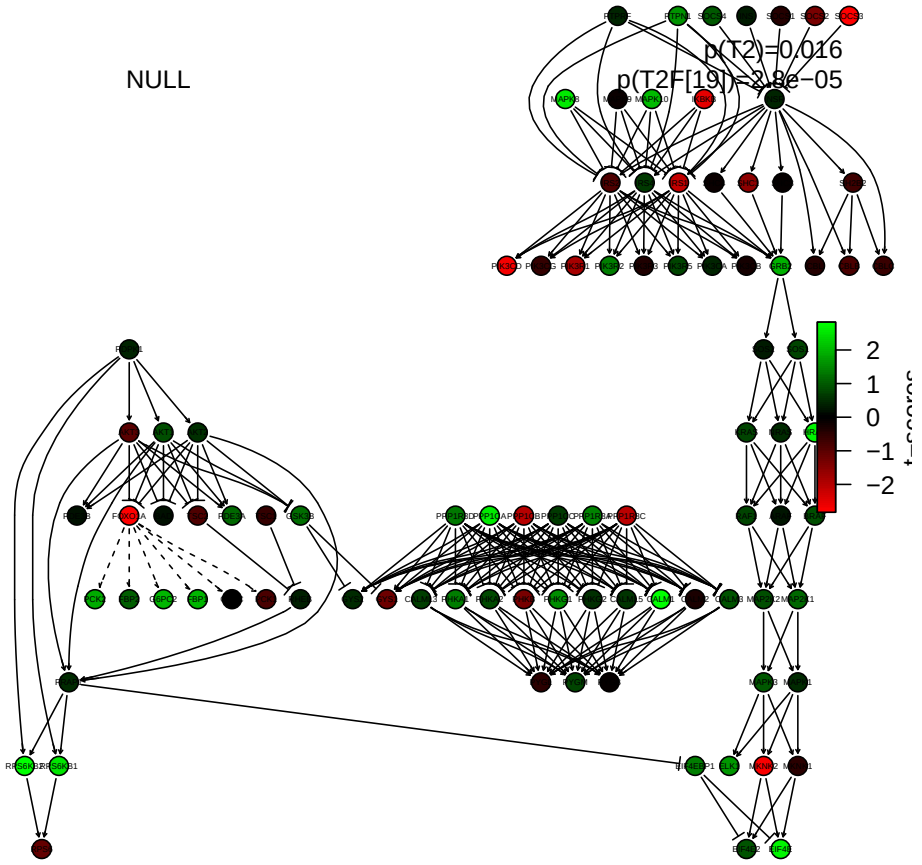

Figure 1: Pathway 1.

```
> stext(side=3, pos=1, txt)
> image.plot(legend.only=TRUE, zlim=range(res$breaks), col=pal, legend.shrink=0.3, legend.w
```

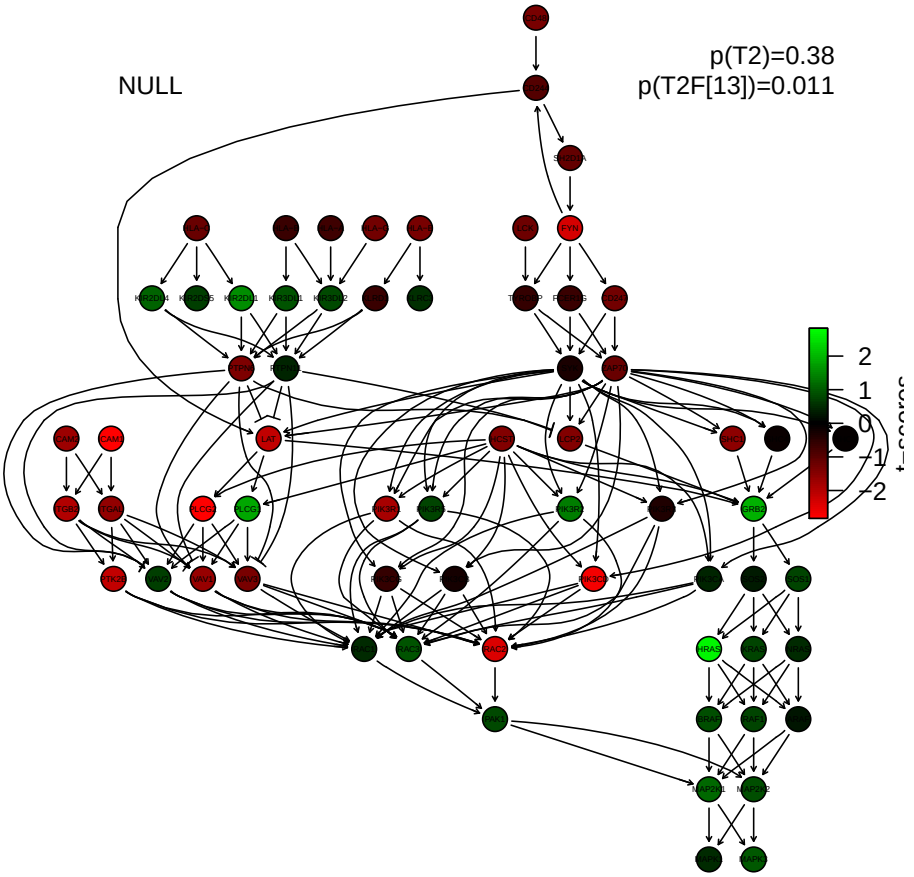

Figure 2: Pathway 2.