Gene set enrichment based on InterPro domain signatures

F. Hahne Tim Beissbarth

May 4, 2016

Abstract

High-throughput technologies like functional screens and gene expression analysis produce extended lists of candidate genes. Gene-Set Enrichment Analysis is a commonly used and well established technique to test for the statistically significant over-representation of particular groups of genes, e.g. pathways. A shortcoming of this method is however, that often genes that are investigated in the experiments have very sparse functional or pathway annotation and therefore cannot be the target of such an analysis. The approach presented here aims to assign lists of genes with limited annotation to previously described functional gene collections or pathways by comparing InterPro domain signatures of the candidate gene lists with domain signatures of gene sets derived from known classifications, e.g. KEGG pathways. The assumption of this approach is that the shared and coordinated function of proteins in a pathway is somehow reflected in their protein domain composition.

1 Introduction

This Vignette shows the typical workflow for a gene set enrichment analysis (GSEA) using the functionality provided in the domainSignatures package. For more information on the method, see (**?**). The first section deals with the special case of testing for enrichment of KEGG pathways, whereas the second section shows how to test for enrichment of arbitrary pathways/selections of genes.

2 KEGG pathway membership

To demonstrate GSEA based on KEGG pathway membership, we use a sample data set consisting of 30 randomly sampled genes and 10 genes that are known to be annotated to a KEGG pathway, say the Notch signaling pathway (hsa04330). The KEGG annotation information is taken directly from the KEGG package, and we are only interested in human pathways for which the indentifier starts with the prefix *hsa*.

```
> library(domainsignatures)
> keggMap <- mget(grep("^hsa", ls(KEGGPATHID2EXTID), value=TRUE),
+                 KEGGPATHID2EXTID)
> univGenes <-  unique(unlist(keggMap))
> randGenes <- sample(univGenes, 30)
> pwLength <- listLen(keggMap)
> randPw <- "hsa04330"
```

```
> set.seed(123)
> pwGenes <- sample(keggMap[[randPw]], 10)
```

In order to perform the analysis, we first need to create an object of class *ipDataSource*. Objects of this class hold the mapping of EntrezGene IDs to the annotation of interest. Basically, this is a list where each list item is a collection of genes by some criterion, e.g. a single pathway. For each of these list items, the mapping to InterPro domains is stored along with the gene identifiers.

There is a convenience function `getKEGGdata` that helps to create *ipDataSource* objects to test for overrepresentation of KEGG pathways. The function will access the Ensembl data base via the biomaRt interface to retrieve the necessary information. Note that you need to pass a vector of all EntrezGene IDs for all genes that were probed in the initial experiment to the function. We call this the "universe" gene set, and in a micro array experiment this would be a list of all genes for which there are probes on the chip. In a cell-based functional assay, this could for instance be a list of all genes that are targeted by a siRNA library. In our example, the universe are all genes that are mapped to any of the currently 229 KEGG pathways.

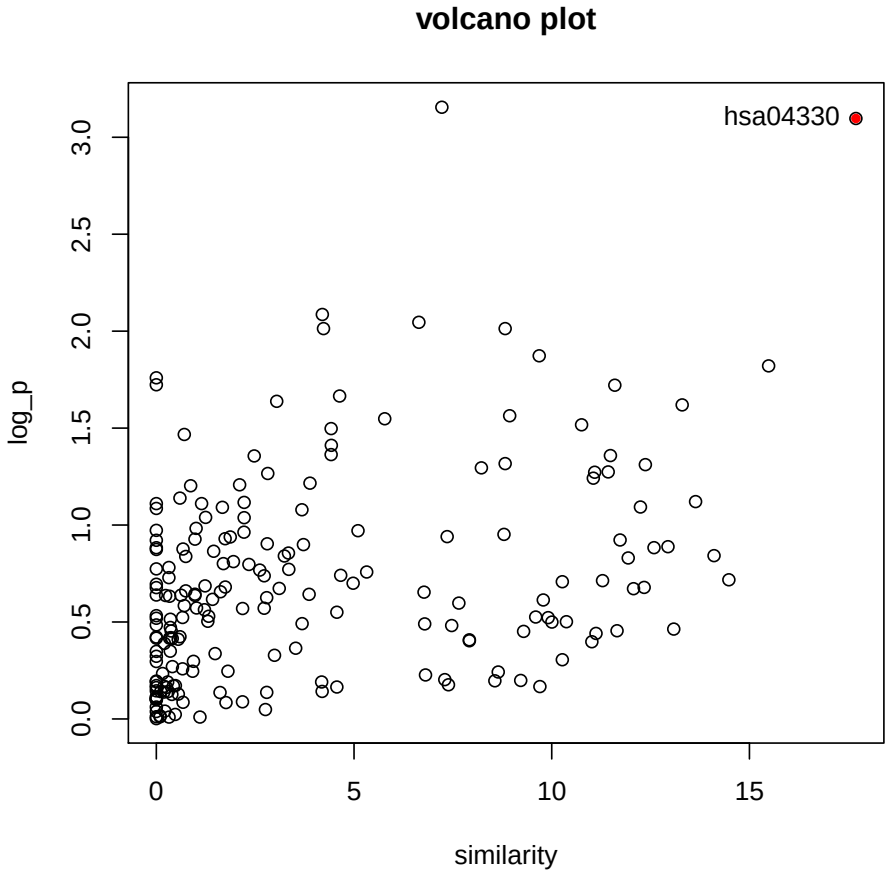
```
> geneset <- c(randGenes, pwGenes)
> universe <- getKEGGdata(univGenes)
> universe

object of class 'ipDataSource' containing mapping data
for type 'KEGG' pathways
   229 pathways with 5894 genes
   and 4909 annotated InterPro domains
```

In a second step we will now test for enrichment by calling the `gseDomain` function. Necessary arguments are `dataSource`, the *ipDataSource* object we just created, and `geneset`, a character vector of EntrezGene IDs for you gene set, in our case the sampled genes. For the sake of performance you can control the number of resampling iterations via the additional argument `n`. For the sake of perfomance, we set `n` to a relatively small value of `1000`. For more reliable results and to reduce the false positive rate, you should use at least 10,000 iterations (the default for `n`).

```
> res <- gseDomain(dataSource=universe, geneset=geneset, n=1000)
```

Let's plot the results in a volcano plot, that is the size of our test statistic against the computed probability. For the x-axis we choose the similarity measure of the gene list's domain signature to the total pathway signature (`res$similarity`) and for the y-axis the negative log transform of the p-value (`res$pvalue`), both computed by `gseDomain`.

```
> volcData <- cbind(similarity=res$similarity, log_p=-log10(res$pvalue))
> plot(volcData, main="volcano plot")
> sel <- which.max(res$similarity)
> points(volcData[sel,,drop=FALSE], pch=20, col="red")
> text(volcData[sel,,drop=FALSE], names(sel), pos=2)
```



We find one pathway which is both significantly enriched and which shows a high similarity score: the Notch signaling pathway we chose at the beginning of this vignette.

```
> names(sel) == randPw
```

```
[1] TRUE
```

```
> getKEGGdescription(randPw)
```

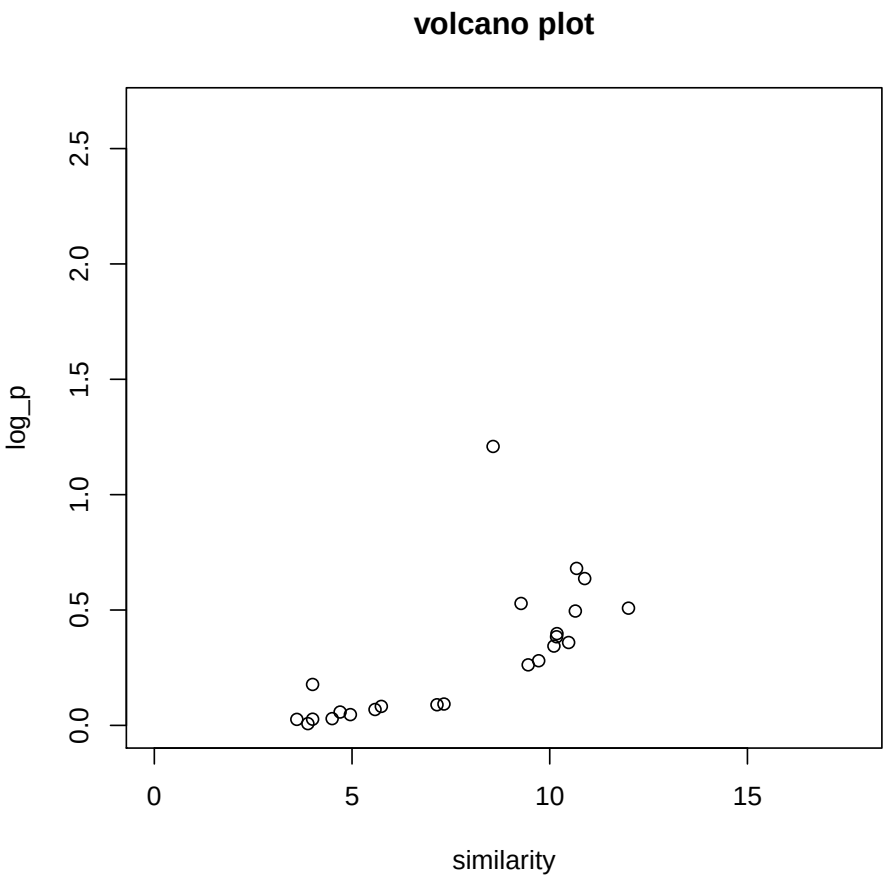
```
            ID             description
04330 hsa04330 Notch signaling pathway
```

There seem to be additional pathways with significant or close to significatn p-values and a lower similarity score; those could either be false positives, pathways that actually contain hsa04330 (there is a certain amount of hierarchy and thus overlap in KEGG), or pathways that were randomly picked up in the initial sampling of the 30 "gene set" genes. Also note that there are extremely sparse or uniform KEGG pathways which this method picks up unreliably.

3 Arbitrary pathways

When not testing for KEGG pathway enrichment but instead for arbitrary pathways or groupings, the single step that is different from the above procedure is the creation of the *ipDataSource* object. You should use the function `dataSource` for that purpose. The only necessary argument is `mapping`, which is a named list that provides the mapping between EntrezGene identifiers and the groupings of genes, where the names of the list items correspond to gene names. The function will automatically fetch the necessary InterPro domain annotation from Ensembl via the BiomaRt interface. For example, we could group our universe genes from the last section according to chromosomes. There is no reason to believe that genes on the same chromosome share extended numbers of common protein domains, so we don't really expect the method to give very useful results here.

```
> ensembl <- useMart("ensembl",dataset="hsapiens_gene_ensembl")
> cMap <- getBM(attributes=c("entrezgene", "chromosome_name"), bmHeader=FALSE,
+               filters="entrezgene", value=univGenes, mart=ensembl)
> cMap <- cMap[cMap$chromosome_name %in% c(as.character(1:23), "X", "Y"),]
> grouping <- split(cMap$chromosome_name, cMap$entrezgene)
> universe2 <- dataSource(grouping)
> geneset2 <- geneset[geneset %in% names(grouping)]

> ## Again, the object was precomputed for perfomance reasons
> res2 <- gseDomain(universe2, geneset2)

> volcData2 <- cbind(similarity=res2$similarity, log_p=-log10(res2$pvalue))
> plot(volcData2, main="volcano plot",
+      xlim=c(0,max(unlist(res$similarity))))
```



As expected, we don't find any of the chromosomes to be over-represented in our gene list.