The *GOSim* package

Holger Fröhlich

January 14, 2010

1 Introduction

The Gene Ontology (GO) has become one of the most widespread systems for systematically annotating gene products within the bioinformatics community and is developed by the Gene Ontology Consortium (22). It is specifically intended for describing gene products with a controlled and structured vocabulary. GO terms are part of a Directed Acyclic Graph (DAG), covering three orthogonal taxonomies or "aspects": *molecular function, biological process* and *cellular component*. Two different kinds of relationship between GO terms exist: the "is-a" relationship and the "part-of" relationship. Providing a standard vocabulary across any biological resources, the GO enables researchers to use this information for automated data analysis.

The *GOSim* package (6) provides the researcher with various information theoretic similarity concepts for GO terms (16; 17; 11; 8; 12; 3; 4). Moreover, since version 1.1.5 *GOSim* contains several new similarity concepts, which are based on so-called diffusion kernel techniques (9). Additionally *GOSim* implements different methods for computing functional similarities between gene products based on the similarties between the associated GO terms (21; 7; 19; 9; 5). This can, for instances, be used for clustering genes according to their biological function (21; 7) and thus may help to get a better understanding of the biological aspects covered by a set of genes.

Since version 1.1 *GOSim* additionally offers the possibility of a GO enrichment analysis using the topGO package (1). Hence, *GOSim* acts now as an umbrella for different analysis methods employing the GO structure.

2 Usage of *GOSim*

To elucidate the usage of *GOSim* we show an example workflow and explain the employed similarity concepts. We create a character vector of Entrez gene IDs, which we assume to be from human:

```
> library(GOSim)
> genes = c("207", "208", "596", "901", "780", "3169", "9518",
+     "2852", "26353", "8614", "7494")
```

Next we investigate the GO annotation within the current ontology (which is *biological process* by default):

```
> getGOInfo(genes)
```

2.1 Term Similarities

Let us examine the similarity of the GO terms for genes "8614" and "2852" in greater detail:

```
> getTermSim(c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186"), method = "Resnik", verbose = FALSE)
```

```
           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166  0.2862381  0.1850912  0.0000000  0.1974792  0.2862381
GO:0007267  0.1850912  0.3933669  0.0000000  0.1850912  0.1850912
GO:0007584  0.0000000  0.0000000  0.5993206  0.0000000  0.0000000
GO:0007165  0.1974792  0.1850912  0.0000000  0.1974792  0.1974792
GO:0007186  0.2862381  0.1850912  0.0000000  0.1974792  0.3466061
```

This calculates Resnik's pairwise similarity between GO terms (16; 17):

$$sim(t, t') = IC_{ms}(t, t') := \max_{\hat{t} \in Pa(t,t')} IC(\hat{t}) \tag{1}$$

Here $Pa(t, t')$ denotes the set of all common ancestors of GO terms t and t', while $IC(t)$ denotes the information content of term t. It is defined as (e.g. (12))

$$IC(\hat{t}) = -\log P(\hat{t}) \tag{2}$$

i.e. as the negative logarithm of the probability of observing $\hat{t}$. The information content of each GO term is already precomputed for each ontology based on the empirical observation, how many times a specific GO term or any of its direct or indirect offsprings appear in the annotation of the GO with gene products.

```
> data("ICsBPhumanall")
> IC[c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186")]
```

```
GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
  3.093797   4.251696   6.477741   2.134449   3.746283
```

This loads the information contents of all GO terms within "biological process". Likewise, the data files `ICsMFhumanall` and `ICsCChumanall` contain the information contents of all GO terms within "molecular function" and "cellular component" for human. Since

GOSim version 1.1.4.0 the information content of GO terms relies on the mapping of Entrez gene IDs to GO terms provided by the libraries org.Dm.eg.db (fly), org.Hs.eg.db (human), org.Mm.eg.db (mouse), org.Pf.plasmo.db (malaria), org.Rn.eg.db (rat) and org.Sc.sgd.db (yeast). Additionally, it is possible to pass a user provided mapping via the function `setEvidenceLevel`. Please refer to the manual pages for details. If only GO terms having certain evidence codes should be considered, one must explicitely calculate the corresponding information contents in the function `calcICs`. Again, more information on this function can be found in the manual pages.

For the similarity computation in (Eq.: 1) normalized information contents are used, which are obtained by dividing the raw information contents by its maximal value:

```
> IC[c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186")]/max(IC)
```

To continue our example from above, let us also calculate Jiang and Conrath's pairwise similarity between GO terms, which is the default, for compairson reasons (8):

```
> getTermSim(c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186"), verbose = FALSE)
```

```
           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166 0.24891623 0.09203921  0.0000000  0.1463202 0.22517173
GO:0007267 0.09203921 0.32521888  0.0000000  0.1058657 0.08453053
GO:0007584 0.00000000 0.00000000  0.4508154  0.0000000 0.00000000
GO:0007165 0.14632024 0.10586565  0.0000000  0.1792028 0.13008555
GO:0007186 0.22517173 0.08453053  0.0000000  0.1300855 0.29291619
```

Jiang and Conrath's similarity measure is defined as

$$sim(t, t') = 1 - \min(1, IC(t) - 2IC_{ms}(t, t') + IC(t')) \tag{3}$$

i.e. the similarity between t and t' is 0, if their normalized distance is at least 1.

Likewise, we can also compute Lin's pairwise similarity between GO terms (11):

```
> getTermSim(c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186"), method = "Lin", verbose = FALSE)
```

```
           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166  1.0000000  0.5447024          0  0.8165066  0.9046085
GO:0007267  0.5447024  1.0000000          0  0.6265295  0.5002649
GO:0007584  0.0000000  0.0000000          1  0.0000000  0.0000000
GO:0007165  0.8165066  0.6265295          0  1.0000000  0.7259126
GO:0007186  0.9046085  0.5002649          0  0.7259126  1.0000000
```

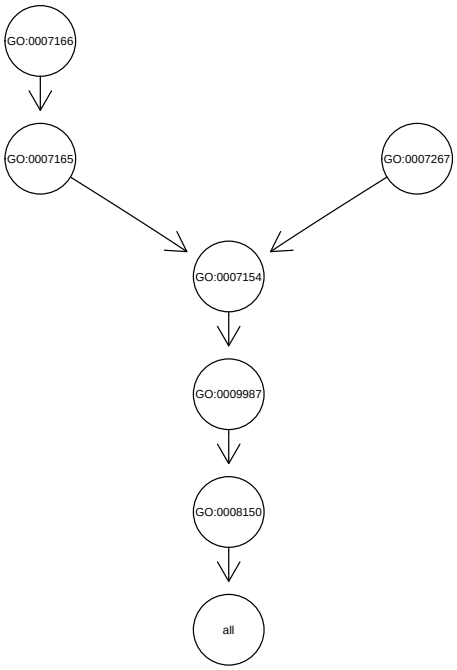

Figure 1: Example of a GO graph starting with leaves GO:0007166 and GO:0007267.

It is defined as:

$$sim(t, t') = \frac{2IC_{ms}(t, t')}{IC(t) + IC(t')} \tag{4}$$

Resnik's, Jiang-Conraths's and Lin's term similarities all refer to $IC_{ms}(t, t')$, the information content of the minimum subsumer of t and t', i.e. of the lowest common ancestor in the hierarchy. For illustration let us plot the GO graph with leaves GO:0007166 and GO:0007267 and let us compute their minimum subsumer (see Fig. 1):

```
> library(igraph)
> G = getGOGraph(c("GO:0007166", "GO:0007267"))
> plot(igraph.from.graphNEL(G))

> getMinimumSubsumer("GO:0007166", "GO:0007267")

[1] "GO:0007154"
```

In contrast to the above defined similarity measures Couto et al. (4) introduced a concept, which is not based on the minimum subsumer, but on the set of all disjunctive common ancestors. Roughly speaking, the idea is not to consider the common ancestor having the highest information content only, but also others, if they are somehow "separate" from each other, i.e. there is a path to t and t' not passing any other of the disjunctive common ancestors.

```
> getDisjCommAnc("GO:0007166", "GO:0007267")

[1] "GO:0007154" "GO:0009987"
```

In this case the set of disjunctive common ancestors only consists of the minimum subsumer, because any path from the other ancestors to GO:0007166 and GO:0007267 would have to pass the minimum subsumer (see Fig. 1).

Based on the notion of disjunctive common ancestors Resnik's similarity concept can be extended by defining:

$$sim(t, t') = IC_{share}(t, t') = \frac{1}{|DisjCommAnc|} \sum_{t \in DisjCommAnc} IC(t) \tag{5}$$

Likewise, Jiang-Conraths's and Lin's measures can be extended as well by replacing $IC_{ms}(t, t')$ by $IC_{share}(t, t')$.

```
> getTermSim(c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186"), method = "CoutoResnik", verbose = FALSE)

           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166  0.2862381  0.1143880  0.0000000  0.1161896  0.1974792
GO:0007267  0.1143880  0.3933669  0.0000000  0.1143880  0.1143880
GO:0007584  0.0000000  0.0000000  0.5993206  0.0000000  0.0000000
GO:0007165  0.1161896  0.1143880  0.0000000  0.1974792  0.1161896
GO:0007186  0.1974792  0.1143880  0.0000000  0.1161896  0.3466061
```

Finally, it should be mentioned that also the depth and density enriched term similarity by Couto et al. (3) has been integrated into *GOSim*:

```
> setEnrichmentFactors(alpha = 0.5, beta = 0.3)
> getTermSim(c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186"), method = "CoutoEnriched", verbose = FALSE)

           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166 0.08193225 0.09392804         NA 0.05378484 0.09574040
GO:0007267 0.09392804 0.15473753         NA 0.06856927 0.10896687
GO:0007584         NA         NA  0.3591852         NA         NA
GO:0007165 0.05378484 0.06856927         NA 0.03899802 0.06273313
GO:0007186 0.09574040 0.10896687         NA 0.06273313 0.12013577
```

Since version 1.1.5 *GOSim* contains several new similarity concepts, which are based on so-called diffusion kernel techniques (9) rather than on the information theoretic ideas presented before. For using these similarity measures it is necessary to pre-compute a diffusion kernel on the Gene Ontology graph, e.g. via

```
> calc.diffusion.kernel(method = "diffKernelLLE")
```

The resulting kernel/similarity matrix is stored in a file called e.g. 'diffKernelLLEB-Phumanall.rda' (meaning local linear embedding diffusion kernel for ontology BP in human using all evidence codes) in the current working directory. The file has then to be moved to GOSim's 'data' directory. Please refer to the manual pages for more detailed information. Once the kernel is created, it has to be loaded into the environment first. Then similarities can be calculated:

```
> load.diffusion.kernel(method = "diffKernelLapl")
> getTermSim(c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186"), method = "diffKernel", verbose = FALSE)
```

Since version 1.2 *GOSim* also contain Schlicker et al.'s and Li et al.'s GO term similarity measures (19; 10), which are both an adaption of Lin's similarity measure. Moreover, the graph information content similarity by Pesquita et al. has been implemented (15).

```
> getTermSim(c("GO:0007166", "GO:0007267", "GO:0007584", "GO:0007165",
+     "GO:0007186"), method = "relevance", verbose = FALSE)
```

```
           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166 0.24891623 0.09203921  0.0000000  0.1463202 0.22517173
GO:0007267 0.09203921 0.32521888  0.0000000  0.1058657 0.08453053
GO:0007584 0.00000000 0.00000000  0.4508154  0.0000000 0.00000000
GO:0007165 0.14632024 0.10586565  0.0000000  0.1792028 0.13008555
GO:0007186 0.22517173 0.08453053  0.0000000  0.1300855 0.29291619
```

2.2 Functional Gene Similarities

The special strength of *GOSim* lies in the possibility not only to calculate similarities for individual GO terms, but also for genes based on their complete GO anntation. Since *GOSim* version 1.1.5 for this purpose the following ideas have been implemented:

1. Maximum and average pairwise GO term similarity
2. Average of best matching GO term similarities (19).
3. Computation of a so-called *optimal assignment* of terms from one gene to those of another one (7).
4. Similarity derived from Hausdorff distances between sets (5).
5. Embedding of each gene into a feature space: (21; 7) proposed to define feature vectors by a gene's maximum GO term similarity to certain prototype genes. More simple (but probably also less accurate), (14) recently proposed to represent each

gene by a feature vector describing the presence/absence of all GO terms. The absence of each GO term is additionally weighted by its information content. Within a feature space gene functional similarities naturally arise as dot products between feature vectors. These dot products can be understood as so-called *kernel functions* (20), as used in e.g. Support Vector Machines (2). Depending on the choice of later normalization (see below) one can arrive at the cosine similarity (Eq.˜6), at the Tanimoto coefficient (Eq.˜7) or at a measure similiar to Lin's one (Eq.˜8, Eq.˜4).

2.2.1 Normalization of Similarities

Often, people want to normalize similarities, e.g. on the interval $[0, 1]$, for better interpretation. To do so, we can perform the transformation

$$sim_{gene}(g, g') \leftarrow \frac{sim_{gene}(g, g')}{\sqrt{sim_{gene}(g, g) sim_{gene}(g', g')}} \tag{6}$$

Provided $sim_{gene} \geq 0$, the consequence will be a similarity of 1 for g with itself and between 0 and 1 for g with any other gene. In case of a feature space embedding this transformation is equivalent to computing the cosine similarity between two feature vectors.

Another possibility is to use Lin's normalization (see Eq. 4):

$$sim_{gene}(g, g') \leftarrow \frac{2sim_{gene}(g, g')}{sim_{gene}(g, g) + sim_{gene}(g', g')} \tag{7}$$

Furthermore, one can use a normalization in the spirit of the Tanimoto coefficient:

$$sim_{gene}(g, g') \leftarrow \frac{sim_{gene}(g, g')}{sim_{gene}(g, g) + sim_{gene}(g', g') - sim_{gene}(g, g')} \tag{8}$$

In case of a feature space embedding the transformation corresponds exactly to the Tanimoto coefficient betweem two feature vectors.

We now give a more detailed overview over the different similarity concepts mentioned above.

2.2.2 Maximum and Average Pairwise GO Term Similarity

The idea of the maximum pairwise GO term similarity is straight forward. Given two genes g and g' annotated with GO terms $t_1, ..., t_n$ and $t'_1, ..., t'_m$ we define the functional similarity between between g and g' as

$$sim_{gene}(g, g') = \max_{\substack{i = 1, , ..., n \\ j = 1, ..., m}} sim(t_i, t'_j) \tag{9}$$

where sim is some similarity measure to compare GO terms t_i and t'_j. Instead of computing the maximum pairwise GO term similarity one may also take the average here.

2.2.3 Average of Best Matching GO Term Similarities

The idea of this approach (19) is to assign each GO term t_i occuring in gene g to its best matching partner $t'_{\pi i}$ in gene g'. Hence multiple GO terms from gene g can be assigned to one GO term from gene g'. A similarity score is computed by taking the average similarity of assigned GO terms. Since, however, genes can have an unequal number of GO terms the result depends on whether GO terms of gene g are assigned to those of gene g' or vice versa. Hence, in (19) it was proposed to either take the maximum or the average of both similarity scores. Both strategies are implemented in *GOSim*.

2.2.4 Optimal Assignment Gene Similarities

To elucidate the idea of the optimal assignment (7), consider the GO terms associated with gene "8614" on one hand and gene "2852" on the other hand:

```
> getGOInfo(c("8614", "2852"))
```

```
           8614        2852
go_id      Character,3 Character,2
Term       Character,3 Character,2
Definition Character,3 Character,2
IC         Numeric,3   Numeric,2
```

Given a similarity concept sim to compare individual GO terms, the idea is now to assign each term of the gene having fewer annotation to exactly one term of the other gene such that the overall similarity is maximized. More formally the optimal assignment problem can be stated as follows: Let π be some permutation of either an n-subset of natural numbers $\{1, ..., m\}$ or an m-subset of natural numbers $\{1, ..., n\}$ (this will be clear from context). Then we are looking for the quantity

$$sim_{gene}(g, g') = \begin{cases} \max_\pi \sum_{i=1}^{n} sim(t_i, t'_{\pi(i)}) & \text{if } m > n \\ \max_\pi \sum_{j=1}^{m} sim(t_{\pi(j)}, t'_j) & \text{otherwise} \end{cases} \tag{10}$$

The computation of (10) corresponds to the solution of the classical maximum weighted bipartite matching (optimal assignment) problem in graph theory and can be carried out in $O(\max(n, m)^3)$ time (13). To prevent that larger lists of terms automatically achieve a higher similarity we may further sim_{gene} divide 10 by $\max(m, n)$.

In our example, using Lin's GO term similarity measure the following assignments are found:

$$\texttt{GO:0007165} \rightarrow \texttt{GO:0007267} \tag{11}$$

$$\texttt{GO:0007186} \rightarrow \texttt{GO:0007166} \tag{12}$$

The resulting similarity matrix is:

```
> getGeneSim(c("8614", "2852"), similarity = "OA", similarityTerm = "Lin",
+     verbose = FALSE)

filtering out genes not mapping to the currently set GO category ... ===> list of  2
          8614      2852
8614 1.0000000 0.5103793
2852 0.5103793 1.0000000
```

Note the difference to a gene similarity that is just based on the maximum GO term similarity and to a gene similarity that is based on the average of best matching GO terms:

```
> getGeneSim(c("8614", "2852"), similarity = "max", similarityTerm = "Lin",
+     verbose = FALSE)

filtering out genes not mapping to the currently set GO category ... ===> list of  2
          8614      2852
8614 1.0000000 0.9046085
2852 0.9046085 1.0000000

> getGeneSim(c("8614", "2852"), similarity = "funSimMax", similarityTerm = "Lin",
+     verbose = FALSE)

filtering out genes not mapping to the currently set GO category ... ===> list of  2
          8614      2852
8614 1.0000000 0.8605575
2852 0.8605575 1.0000000
```

2.2.5 Gene Similarities In the Spirit of Hausdorff Metrics

Hausdorff metrics are a general concept for measuring distances between compact subsets of a metric space. Let X and Y be the two sets of GO terms associated to genes g and g', and let $d(t, t')$ denote the distanc between GO terms t and t'. Then the Hausdorff distance X and Y is defined as

$$d_{Hausdorff}(X, Y) = \max\{\sup_{t \in X} \inf_{t' \in Y} d(t, t'), \sup_{t' \in Y} \inf_{t \in X} d(t, t')\} \tag{13}$$

Using Hausdorff metrics for measuring gene functional distances was proposed in (5). We translate the idea to define a similarity measure between g and g':

$$sim_{gene}(g, g') = \min\{\min_{i=1,\ldots,n} \max_{j=1,\ldots,m} sim(t_i, t'_j), \min_{j=1,\ldots,m} \max_{i=1,\ldots,n} sim(t_i, t'_j)\} \tag{14}$$

```
> getGeneSim(c("8614", "2852"), similarity = "hausdorff", similarityTerm = "Lin",
+     verbose = FALSE)

filtering out genes not mapping to the currently set GO category ... ===> list of  2
     8614 2852
8614    1    0
2852    0    1
```

2.2.6 Feature Space Embedding of Gene Products

The Simple Approach (14) proposed to represent each gene by a feature vector describing the presence/absence of all GO terms. The absence of each GO term is additionally weighted by its information content. In the feature space similarities arise as dot products. Hence, the similarity between two GO terms t and t' is implicitly defined as the product of their information content values, hence igoring the exact DAG structure of the Gene Ontology as employed by the GO term similarity measures explained in the beginning of this document.

```
> getGeneSim(c("8614", "2852"), similarity = "dot", method = "Tanimoto",
+     verbose = FALSE)

filtering out genes not mapping to the currently set GO category ... ===> list of  2
          8614      2852
8614 1.0000000 0.1131520
2852 0.1131520 1.0000000
```

This will calculate the Tanimoto coefficient between feature vectors as a similarity measure. It is possible to retrieve the feature vectors via:

```
> features = getGeneFeatures(c("8614", "2852"))
```

Embeddings via GO Term Similarities to Prototype Genes This approach is due to (21; 7). The idea is to define a feature vector for each gene by its pairwise GO term similarity to certain prototype genes, i.e. the prototype genes form a (nonorthogonal) basis, and each gene is defined relative to this basis. The prototype genes can eithed be defined a priori or one can use one of the heuristics implemented in the function `selectPrototypes`. The default behavior is to select the 250 best annotated genes, i.e. which have been annotated with GO terms most often:

```
> proto = selectPrototypes(verbose = FALSE)
```

We now calculate for each gene g feature vectors $\phi(g)$ by using their similarity to all prototypes $p_1, ..., p_n$:

$$\phi(g) = (sim'(g, p_1), ..., sim'(g, p_n))^T \tag{15}$$

Here sim' by default is the maximum pairwise GO term similarity. Alternatively, one can use other similarity measures for sim' as well. These similarity measures can by itself again be combined with arbitrary GO term similarity concepts. The default is the Jiang-Conrath term similarity.

Because the feature vectors are very high-dimensional we usually perform a principal component analysis (PCA) to project the data into a lower dimensional subspace:

```
> PHI = getGeneFeaturesPrototypes(genes, prototypes = proto, verbose = FALSE)
```

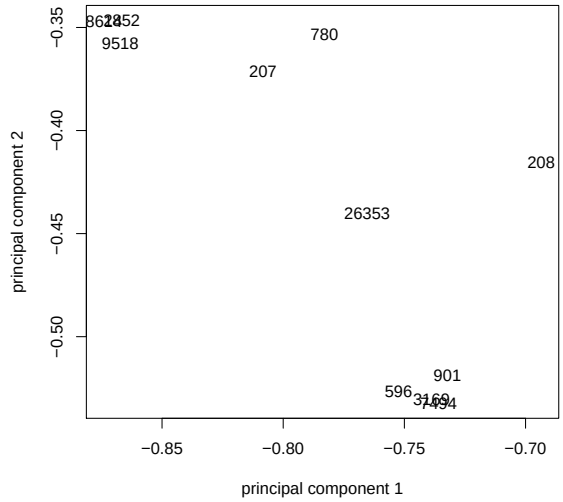

Figure 2: Embedding of the genes into the feature space spanned by the first 2 principal components

This uses the above define prototypes to calculate feature vectors and performs a PCA afterwards. The number of principal components is chosen such that at least 95% of the total variance in feature space can be explained (this is a relatively conservatve criterion).

We can now plot our genes in the space spanned by the first 2 principal components to get an impression of the relative "position" of the genes to each other in the feature space (see Fig. 2). The feature vectors are normalized to Euclidian norm 1 by default:

```
> x = seq(min(PHI$features[, 1]), max(PHI$features[, 1]), length.out = 100)
> y = seq(min(PHI$features[, 2]), max(PHI$features[, 2]), length.out = 100)
> plot(x, y, xlab = "principal component 1", ylab = "principal component 2",
+     type = "n")
> text(PHI$features[, 1], PHI$features[, 2], labels = genes)
```

Finally, we can directly calculate the similarities of the genes to each other, this time using the Resnik's GO term similarity concept. These similarities may then be used to cluster genes with respect to their function:

```
> sim = getGeneSimPrototypes(genes, prototypes = proto, similarityTerm = "Resnik",
+     verbose = FALSE)
> h = hclust(as.dist(1 - sim$similarity), "ward")
> plot(h, xlab = "")
```

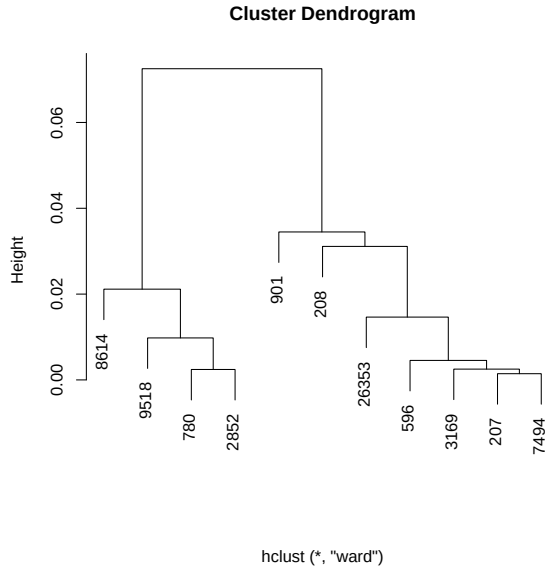

Figure 3: Possible functional clustering of the genes using Ward's method.

This produces a hierarchical clustering of all genes using Ward's method (see Fig. 3).

2.2.7 Combination of Similarities from Different Ontologies

It should be mentioned that up to now all similarity computations were performed within the ontology "biological process". One could imagine to combine functional similarities between gene products with regard to different taxonomies. An obvious way for doing so would be to consider the sum of the respective similarities:

$$sim_{total}(g, g') = sim_{Ontology1}(g, g') + sim_{Ontology2}(g, g') \tag{16}$$

Of course, one could also use a weighted averaging scheme here, if desired.

2.3 Cluster Evaluations

GOSim has the possibility to evaluate a given clustering of genes or terms by means of their GO similarities. Supposed, based on other experiments (e.g. microarry), we have decided to put genes "8614", "9518", "780", "2852" in one group, genes "3169", "207", "7494", "596" in a second and the rest in a third group. Then we can ask ourselves, how similar these groups are with respect to their GO annotations:

```
> ev = evaluateClustering(c(2, 3, 2, 3, 1, 2, 1, 1, 3, 1, 2), sim$similarity)
> plot(ev$clustersil, main = "")
```

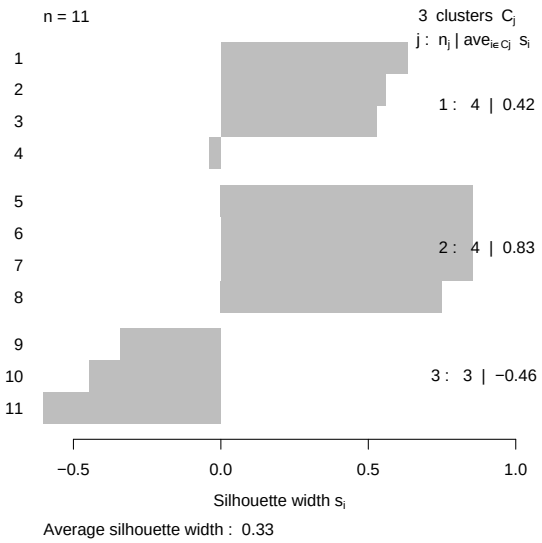

Figure 4: Silhouette plot of a possible given grouping of genes.

A good indiciation of the clustering qualitiy can be obtained by looking at the cluster silhouettes (18) (see Fig. 4). This shows that clusters 1 and 2 are relatively homogenous with respect to the functional similarity of the genes contained in it, while the genes in cluster 3 are more dissimilar.

2.4 GO Enrichment Analysis

Since version 1.1 *GOSim* also offers the possibility of a GO enrichemnt analysis. Suppose, we may now want to get a clearer picture of the genes involved in cluster 1. For this purpose we use the topGO tool (1).

```
> gomap <- get("gomap", env = GOSimEnv)
> allgenes = unique(c(sample(names(gomap), 1000), genes))
> GOenrichment(c("8614", "9518", "780", "2852"), allgenes)

Building most specific GOs .....        ( 1144 GO terms found. )

Build GO DAG topology ..........        ( 2914 GO terms and 5933 relations. )

Annotating nodes ...............        ( 809 genes annotated to the GO terms. )

                        -- Elim Algorithm --
```

```
                the algorithm is scoring 38 nontrivial nodes
                parameters:
                        test statistic:  Fisher test
                        cutOff:  0.01

        Level 9:        2 nodes to be scored        (0 eliminated genes)

        Level 8:        3 nodes to be scored        (0 eliminated genes)

        Level 7:        3 nodes to be scored        (0 eliminated genes)

        Level 6:        4 nodes to be scored        (0 eliminated genes)

        Level 5:        6 nodes to be scored        (101 eliminated genes)

        Level 4:        6 nodes to be scored        (101 eliminated genes)

        Level 3:        8 nodes to be scored        (101 eliminated genes)

        Level 2:        5 nodes to be scored        (101 eliminated genes)

        Level 1:        1 nodes to be scored        (101 eliminated genes)
$GOTerms
          go_id                                                Term
15456 GO:0007166 cell surface receptor linked signal transduction

15456 Any series of molecular signals initiated by the binding of an extracellular l

$p.values
  GO:0007166
0.0002304699

$genes
$genes$`GO:0007166`
  [1] "10203"  "10343"  "10580"  "10603"  "10637"  "10887"  "116535" "119774"
  [9] "120065" "120066" "1232"   "168620" "1855"   "1856"   "186"    "1889"
 [17] "1950"   "2041"   "2046"   "207"    "208"    "2155"   "219428" "2249"
 [25] "2260"   "23235"  "2331"   "23529"  "2566"   "2570"   "2587"   "26211"
 [33] "26219"  "26494"  "26497"  "26658"  "26689"  "2693"   "27239"  "2842"
 [41] "2852"   "2893"   "2931"   "29767"  "2984"   "3175"   "3352"   "3354"
 [49] "3355"   "341418" "342372" "343406" "3667"   "374"    "3824"   "3856"
 [57] "390059" "390061" "390093" "390432" "392309" "399491" "401190" "401992"
```

```
 [65] "4091"   "43847"  "440153" "4615"   "4792"   "4914"   "4986"   "50852"
 [73] "50964"  "51135"  "51554"  "5159"   "54764"  "5494"   "5739"   "655"
 [81] "7049"   "7070"   "7323"   "780"    "799"    "80243"  "81300"  "81309"
 [89] "83550"  "8503"   "8526"   "8614"   "8737"   "8823"   "9209"   "9241"
 [97] "93"     "9370"   "9463"   "9518"   "9934"
```

References

[1] Adrian Alexa, Jörg Rahnenführer, and Thomas Lengauer. Improved scoring of functional groups from gene expression data by decorrelating GO graph structure. *Bioinformatics*, 22(13):1600 – 1607, 2006.

[2] C.˜Cortes and V.˜Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

[3] F.˜Couto, M.˜Silva, and P.˜Coutinho. Implementation of a Functional Semantic Similarity Measure between Gene-Products. Technical Report DI/FCUL TR 03–29, Department of Informatics, University of Lisbon, 2003.

[4] F.˜Couto, M.˜Silva, and P.˜Coutinho. Semantic Similarity over the Gene Ontology: Family Correlation and Selecting Disjunctive Ancestors. In *Conference in Information and Knowledge Management*, 2005.

[5] Angela del Pozo, Florencio Pazos, and Alfonso Valencia. Defining functional distances over gene ontology. *BMC Bioinformatics*, 9:50, 2008.

[6] H.˜Fröhlich, N.˜Speer, A.˜Poustka, and T.˜Beissbarth. GOSim - An R-Package for Computation of Information Theoretic GO Similarities Between Terms and Gene Products. *BMC Bioinformatics*, 8:166, 2007.

[7] H.˜Fröhlich, N.˜Speer, and A.˜Zell. Kernel based functional gene grouping. In *Proc. Int. Joint Conf. Neural Networks*, pages 6886 – 6891, 2006.

[8] J.˜Jiang and D.˜Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, Taiwan, 1998.

[9] Gilad Lerman and Boris˜E Shakhnovich. Defining functional distance using manifold embeddings of gene ontology annotations. *Proc Natl Acad Sci U S A*, 104(27):11334–11339, Jul 2007.

[10] B.˜Li, J.˜Wang, A.˜Feltus, J.˜Zhou, and F.˜Luo. Effectively integrating information content and structural relationship to improve the go-based similarity measure between proteins. *BMC Bioinformatics*, 2009. in press.

[11] D.˜Lin. An information-theoretic definition of similarity. In Morgan Kaufmann, editor, *Proceedings of the 15th International Conference on Machine Learning*, volume˜1, pages 296–304, San Francisco, CA, 1998.

[12] P.W. Lord, R.D. Stevens, A.˜Brass, and C.A. Goble. Semantic similarity measures as tools for exploring the gene ontology. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 601–612, 2003.

[13] K.˜Mehlhorn and S.˜Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[14] Meeta Mistry and Paul Pavlidis. Gene ontology term overlap as a measure of gene functional similarity. *BMC Bioinformatics*, 9:327, 2008.

[15] C.˜Pesquita, D.˜Faria, H.˜Bastos, A.˜Falcao, and F.˜Couto. Evaluating go-based semantic similarity measures. In *Proc. 10th Annual Bio-Ontologies Meeting 2007*, volume 2007, pages 37 – 40, 2007.

[16] P.˜Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume˜1, pages 448–453, Montreal, 1995.

[17] P.˜Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambigiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[18] P.J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comp. and Applied Mathematics*, 20:53–65, 1987.

[19] Andreas Schlicker, Francisco˜S Domingues, Jörg Rahnenführer, and Thomas Lengauer. A new measure for functional similarity of gene products based on Gene Ontology. *BMC Bioinformatics*, 7:302, 2006.

[20] B.˜Schölkopf and A.˜J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[21] N.˜Speer, H.˜Fröhlich, C.˜Spieth, and A.˜Zell. Functional grouping of genes using spectral clustering and gene ontology. In *Proc. Int. Joint Conf. Neural Networks*, pages 298 – 303, 2005.

[22] The Gene Ontology Consortium. The gene ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32:D258–D261, 2004.