The *GOSim* package

Holger Fröhlich

October 17, 2016

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 (**?**). 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 (**?**) provides the researcher with various information theoretic similarity concepts for GO terms (**???????**). Moreover, since version 1.1.5 *GOSim* contains several new similarity concepts, which are based on so-called diffusion kernel techniques (**?**). Additionally *GOSim* implements different methods for computing functional similarities between gene products based on the similarties between the associated GO terms (**?????**). This can, for instances, be used for clustering genes according to their biological function (**??**) 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 (**?**). 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)
```

```
           207            208           596            901          780
go_id      Character,118 Character,30 Character,111 Character,4 Character,17
Term       Character,118 Character,30 Character,111 Character,4 Character,17
Definition Character,118 Character,30 Character,111 Character,4 Character,17
IC         Numeric,118   Numeric,30   Numeric,111   Numeric,4   Numeric,17
           3169          9518         2852          26353       8614
go_id      Character,29 Character,9 Character,55 Character,2 Character,12
Term       Character,29 Character,9 Character,55 Character,2 Character,12
Definition Character,29 Character,9 Character,55 Character,2 Character,12
IC         Numeric,29   Numeric,9   Numeric,55   Numeric,2   Numeric,12
           7494
go_id      Character,55
Term       Character,55
Definition Character,55
IC         Numeric,55
```

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"),me
```

```
           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166  0.2628131  0.1806383  0.1266641  0.1945233  0.1945233
GO:0007267  0.1806383  0.3551639  0.0000000  0.1806383  0.1806383
GO:0007584  0.1266641  0.0000000  0.5128961  0.1266641  0.1266641
GO:0007165  0.1945233  0.1806383  0.1266641  0.1945233  0.1945233
GO:0007186  0.1945233  0.1806383  0.1266641  0.1945233  0.4016432
```

This calculates Resnik's pairwise similarity between GO terms (**??**):

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

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. **?**)

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

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. GOSim provides a normalized version of Resnik's similarity measure, which divides the information content of the minimum subsumber by the maximum information content of all GO terms, hence obtaining a number between 0 and 1.

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

GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
  3.006413   4.062846   5.867200   2.225221   4.594539
```

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 primary gene IDs (mainly Entrez) to GO terms provided by the libraries org.Dm.eg.db (fly), org.Hs.eg.db (human), org.Mm.eg.db (mouse), etc. 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.

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 (**?**):

```
> getTermSim(c("GO:0007166","GO:0007267","GO:0007584","GO:0007165","GO:0007186"),ve

           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166  0.9505312  0.5105747  0.2498911  0.7587689  0.5222505
GO:0007267  0.5105747  0.9828000  0.0000000  0.5740054  0.4169139
GO:0007584  0.2498911  0.0000000  0.9971692  0.2740140  0.2119568
GO:0007165  0.7587689  0.5740054  0.2740140  0.8919565  0.5820734
GO:0007186  0.5222505  0.4169139  0.2119568  0.5820734  0.9898931
```

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 (**?**):

```
> getTermSim(c("GO:0007166","GO:0007267","GO:0007584","GO:0007165","GO:0007186"),me
```

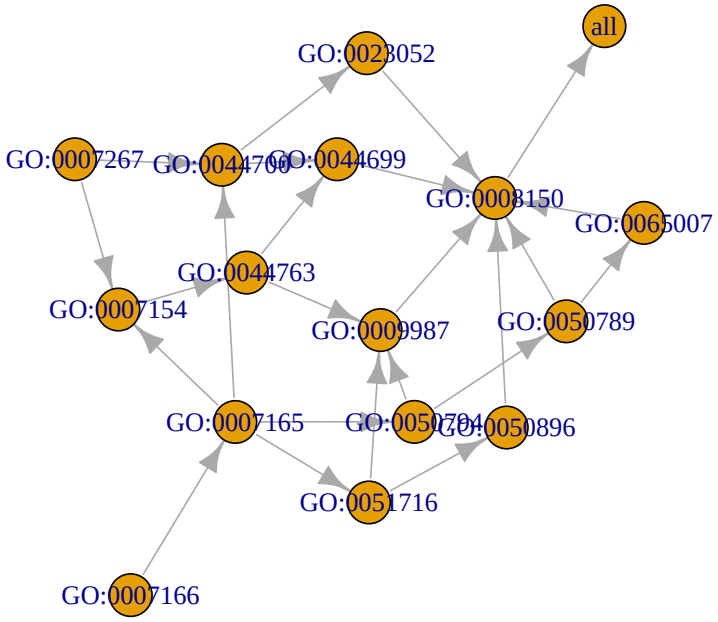
```
           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166  1.0000000  0.5846115  0.3265762  0.8506792  0.5855112
GO:0007267  0.5846115  1.0000000  0.0000000  0.6572401  0.4773693
GO:0007584  0.3265762  0.0000000  1.0000000  0.3581018  0.2770009
GO:0007165  0.8506792  0.6572401  0.3581018  1.0000000  0.6525805
GO:0007186  0.5855112  0.4773693  0.2770009  0.6525805  1.0000000
```

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. **??**):

```
> library(igraph)
> G = getGOGraph(c("GO:0007166","GO:0007267"))
> G2 = igraph.from.graphNEL(G)
> plot(G2, vertex.label=V(G2)$name)
```




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

```
[1] "GO:0023052"
```

In contrast to the above defined similarity measures Couto et al. (**?**) 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 exists a path to t or to t' not passing any other of the disjunctive common ancestors.

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

```
[1] "GO:0007154" "GO:0009987" "GO:0023052" "GO:0044699" "GO:0044700"
[6] "GO:0044763"
```

In this case the set of disjunctive common ancestors consists of the minimum subsumer, GO:0007154, and its parent, GO:0009987, because from both there exists a path to GO:0007166 not passing any other disjunctive common ancestor(see Fig. **??**).

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

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

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"),method="CoutoResnik",verbose=FALSE)
```

```
           GO:0007166 GO:0007267
GO:0007166   3.006413   1.507568
GO:0007267   1.507568   4.062846
```

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

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

```
           GO:0007166 GO:0007267
GO:0007166   9.038517    0.00000
GO:0007267   0.000000   16.50672
```

Since version 1.1.5 *GOSim* contains several new similarity concepts, which are based on so-called diffusion kernel techniques (**?**) 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 via `calc.diffusion.kernel`. This will take some time and result in a kernel/similarity matrix that is stored in a file called e.g. 'diffKernelpowerBPhumanall.rda' (meaning matrix power diffusion kernel for ontology BP in human using all evidence codes) in the current working directory. Once the kernel is created, it has to be loaded into the environment first `load.diffusion.kernel`. Afterwards GO term similarities can be computed via function `getTermSim`. Please check the manual pages for details.

Since version 1.2 *GOSim* also contains Schlicker et al.'s GO term similarity measure (**?**), which is an adaption of Lin's similarity measure. Moreover, the graph information content similarity by Pesquita et al. has been implemented (**?**).

```
> getTermSim(c("GO:0007166","GO:0007267","GO:0007584","GO:0007165","GO:0007186"),me
```

```
           GO:0007166 GO:0007267 GO:0007584 GO:0007165 GO:0007186
GO:0007166  0.9505312  0.5105747  0.2498911  0.7587689  0.5222505
GO:0007267  0.5105747  0.9828000  0.0000000  0.5740054  0.4169139
GO:0007584  0.2498911  0.0000000  0.9971692  0.2740140  0.2119568
GO:0007165  0.7587689  0.5740054  0.2740140  0.8919565  0.5820734
GO:0007186  0.5222505  0.4169139  0.2119568  0.5820734  0.9898931
```

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 (**?**).
3. Computation of a so-called *optimal assignment* of terms from one gene to those of another one (**?**).
4. Similarity derived from Hausdorff distances between sets (**?**).
5. Embedding of each gene into a feature space: (**??**) proposed to define feature vectors by a gene's maximum GO term similarity to certain prototype genes. More simple (but probably also less accurate), (**?**) 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* (**?**), as used in e.g. Support Vector Machines (**?**). Depending on the choice of later normalization (see below) one can arrive at the cosine similarity (Eq. **??**), at the Tanimoto coefficient (Eq. **??**) or at a measure similiar to Lin's one (Eq. **??**, Eq. **??**).

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. **??**):

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

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. This idea is, for instance, realized in FuSSiMeg (**?**). 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 (**?**) 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 **?** 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 (**?**), 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,12 Character,55
Term       Character,12 Character,55
Definition Character,12 Character,55
IC         Numeric,12   Numeric,55
```

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

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

The computation of (**??**) 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 (**?**). To prevent that larger lists of terms automatically achieve a higher similarity we may further sim_{gene} divide **??** by $\max(m, n)$.

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

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

filtering out genes not mapping to the currently set GO category ... ===> list of  2
          8614      2852
8614 1.0000000 0.3108823
2852 0.3108823 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 0.8833252
2852 0.8833252 1.0000000

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

filtering out genes not mapping to the currently set GO category ... ===> list of  2
          8614      2852
8614 1.0000000 0.6597983
2852 0.6597983 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

$$d_{Hausdorff}(X, Y) = \max\{\sup_{t \in X} \inf_{t' \in Y} d(t, t'), \sup_{t' \in Y} \inf_{t \in X} d(t, t')\} \tag{11}$$

Using Hausdorff metrics for measuring gene functional distances was proposed in **?**. We translate the idea to define a similarity measure between g and g' (see the difference to previous GOSim versions):

$$sim_{gene}(g, g') = \exp(-d_{Hausdorff}(g, g')) \tag{12}$$

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

```
filtering out genes not mapping to the currently set GO category ... ===> list of  2
          8614      2852
8614 1.0000000 0.9873622
2852 0.9873622 1.0000000
```

2.2.6 Feature Space Embedding of Gene Products

The Simple Approach **?** 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  NaN
2852  NaN    1
```

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"))
```

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

Embeddings via GO Term Similarities to Prototype Genes This approach is due to **??**. 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, but here we just use 3 for computational reasons:

```
> proto = selectPrototypes(n=3,verbose=FALSE)
```

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

$$\phi(g) = (sim'(g, p_1), ..., sim'(g, p_n))^T \tag{13}$$

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. The results are not shown here due to long computation time.

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

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. **??**). 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[1:3],prototypes=proto,similarityTerm="Resnik",ve
> h=hclust(as.dist(1-sim$similarity),"average")
> plot(h,xlab="")
```

This produces a hierarchical clustering of all genes using average linkage clustering (see Fig. **??**).

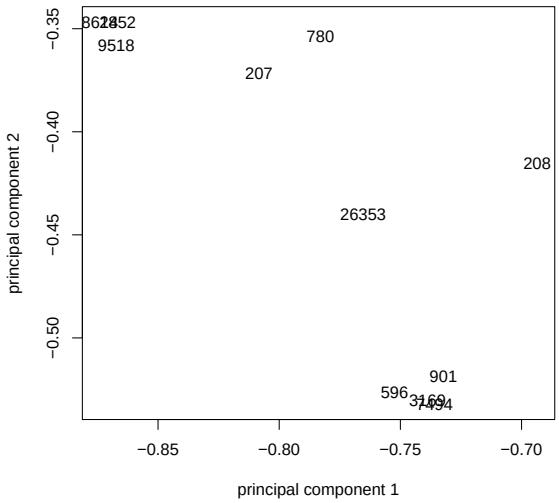


Figure 1: Embedding of genes into feature space spanned by the first 2 principal components

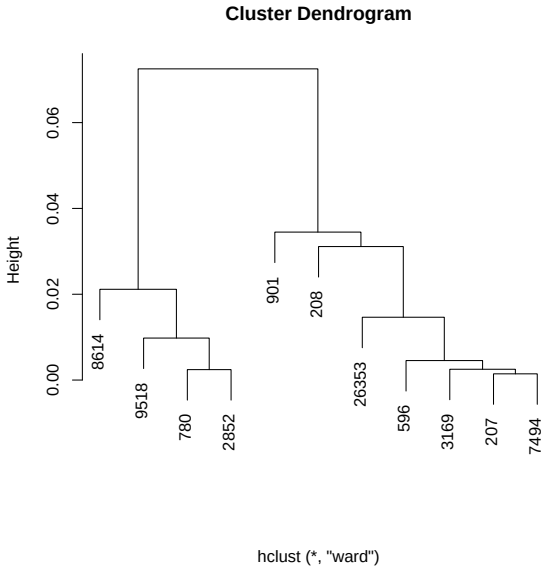


Figure 2: Possible functional clustering of the genes using Ward's method.

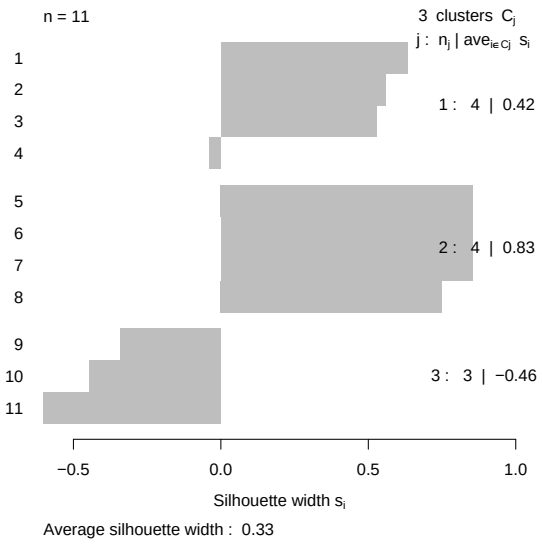


Figure 3: Silhouette plot of a possible given grouping of genes.

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:

$$sim_{total}(g, g') = sim_{Ontology1}(g, g') + sim_{Ontology2}(g, g') \tag{14}$$

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="")
```

A good indiciation of the clustering qualitiy can be obtained by looking at the cluster silhouettes (**?**) (see Fig. **??**). 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 enrichment 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 (**?**).

```
> library(org.Hs.eg.db)
> library(topGO)
> allgenes = union(c("8614", "9518", "780", "2852"), sample(keys(org.Hs.egGO), 100
> GOenrichment(c("8614", "9518", "780", "2852"), allgenes) # print out what cluste

$GOTerms
          go_id                                                Term
16626 GO:0006874                    cellular calcium ion homeostasis
17112 GO:0007167   enzyme linked receptor protein signaling pathway
17799 GO:0007566                                 embryo implantation
18590 GO:0008285          negative regulation of cell proliferation
24595 GO:0010817                       regulation of hormone levels
25063 GO:0014070                response to organic cyclic compound
33463 GO:0022411                     cellular component disassembly
42904 GO:0033993                                  response to lipid
48285 GO:0040008                               regulation of growth
51752 GO:0043408                         regulation of MAPK cascade
51858 GO:0043434                        response to peptide hormone
69933 GO:0051128      regulation of cellular component organization
70070 GO:0051149 positive regulation of muscle cell differentiation
72416 GO:0051924                regulation of calcium ion transport
80046 GO:0071310             cellular response to organic substance

16626
17112           Any series of molecular signals initiated by the binding of an extra
17799
18590
24595           Any process that modulates the levels of hormone within an organism
25063
33463
42904
48285
51752
51858 Any process that results in a change in state or activity of a cell or an orga
69933                                                          Any process that mo
70070
72416
```

```
80046

$p.values
  GO:0033993   GO:0008285   GO:0014070   GO:0040008   GO:0010817   GO:0006874
0.0094302725 0.0094302725 0.0080150606 0.0094302725 0.0044303982 0.0034611203
  GO:0051924   GO:0007566   GO:0051128   GO:0051149   GO:0043434   GO:0022411
0.0018705774 0.0001269089 0.0038207839 0.0003790678 0.0026073100 0.0067089579
  GO:0043408   GO:0071310   GO:0007167
0.0067089579 0.0034717538 0.0007721333

$genes
$genes$`GO:0033993`
 [1] "10002"  "2002"   "2274"   "2852"   "347732" "3779"   "4831"   "51366"
 [9] "54834"  "57402"  "7099"   "85315"  "8614"

$genes$`GO:0008285`
 [1] "10002" "1030"  "10608" "23560" "26272" "2852"  "3635"  "3664"  "5918"
[10] "604"   "675"   "780"   "8850"

$genes$`GO:0014070`
 [1] "10002"  "1119"   "2002"   "2274"   "255189" "2852"   "347732" "3779"
 [9] "51366"  "85315"  "8614"   "9127"

$genes$`GO:0040008`
 [1] "10718"  "1435"   "170302" "3574"   "55636"  "55929"  "5781"   "604"
 [9] "64393"  "780"    "7869"   "8614"   "9798"

$genes$`GO:0010817`
[1] "1584"   "2852"   "286676" "291"    "55636"  "5781"   "7173"   "8614"
[9] "9607"

$genes$`GO:0006874`
[1] "2852"  "287"   "55636" "6288"  "781"   "784"   "7852"  "8614"

$genes$`GO:0051924`
[1] "2852"  "287"   "55636" "781"   "784"   "8614"

$genes$`GO:0007566`
[1] "780"  "8614"

$genes$`GO:0051128`
 [1] "10533"  "10650"  "10718"  "113251" "1289"   "1456"   "157922" "1690"
```

```
 [9] "2041"   "220134" "23768"  "24144"  "2852"   "4286"   "4831"   "51366"
[17] "54543"  "55607"  "55733"  "5781"   "5830"   "604"    "6281"   "6712"
[25] "7099"   "780"    "7869"   "79784"  "79998"  "83700"  "9518"   "9798"

$genes$`GO:0051149`
[1] "2852"  "55662" "9518"

$genes$`GO:0043434`
[1] "1584" "2852" "4831" "5781" "8471" "8614" "8850"

$genes$`GO:0022411`
 [1] "10533" "1504"  "24144" "2852"  "3005"  "54543" "6712"  "780"   "92399"
[10] "9798"  "9801"

$genes$`GO:0043408`
 [1] "112464" "2074"   "2651"   "2852"   "5781"   "6288"   "7099"   "7852"
 [9] "84231"  "9518"   "9607"

$genes$`GO:0071310`
 [1] "10002"  "1030"   "11015"  "1435"   "1584"   "2002"   "2274"   "23768"
 [9] "255189" "26509"  "2651"   "27179"  "27316"  "2852"   "3664"   "3779"
[17] "4831"   "51366"  "51496"  "5434"   "5781"   "63893"  "7099"   "7852"
[25] "80059"  "8471"   "85315"  "8614"   "8809"   "8850"   "9518"

$genes$`GO:0007167`
 [1] "1030"  "10627" "10718" "1435"  "2041"  "23768" "26509" "2651"  "2852"
[10] "51107" "51378" "51496" "5434"  "5781"  "63893" "64096" "780"   "8471"
[19] "9518"
```