Basic GO Usage

R. Gentleman

April 30, 2018

Introduction

In this vignette we describe some of the basic characteristics of the data available from the Gene Ontology (GO), (The Gene Ontology Consortium, 2000) and how these data have been incorporated into Bioconductor. We assume that readers are familiar with the basic DAG structure of GO and with the mappings of genes to GO terms that are provide by GOA (Camon et al., 2004). We consider these basic structures and properties quite briefly.

GO, itself, is a structured terminology. The ontology describes genes and gene products and is divided into three separate ontologies. One for cellular component (CC), one for molecular function (MF) and one for biological process (BP). We maintain those same distinctions were appropriate. The relationship between terms is a parent-child one, where the parents of any term are less specific than the child. The mapping in either direction can be one to many (so a child may have many parents and a parent may have many children). There is a single root node for all ontologies as well as separate root nodes for each of the three ontologies named above. These terms are structured as a directed acyclic graph (or a DAG).

GO itself is only the collection of terms; the descriptions of genes, gene products, what they do, where they do it and so on. But there is no direct association of genes to terms. The assignment of genes to terms is carried out by others, in particular the GOA project (Camon et al., 2004). It is this assignment that makes GO useful for data analysis and hence it is the combined relationship between the structure of the terms and the assignment of genes to terms that is the concern of the *GO.db* package.

The basis for child-parent relationships in GO can be either an *is-a* relationship, where the child term is a more specific version of the parent. Or, it can be a *has-a*, or *part-of* relationship where the child is a part of the parent. For example a telomere is a part-of a chromosome.

Genes are assigned to terms on the basis of their LocusLink ID. For this reason we make most of our mappings and functions work for LocusLink identifiers. Users of specific chips, or data with other gene identifiers should first map their identifiers to LocusLink before using *GOstats*.

A gene is mapped only to the most specific terms that are applicable to it (in each ontology). Then, all less specific terms are also applicable and they are easily obtained by traversing the set of parent relationships down to the root node. In practice many of these mappings are precomputed and easily obtained from the different hash tables provided by the *GO.db* package.

Mapping of a gene to a term can be based on many different things. GO and GOA provide an extensive set of evidence codes, some of which are given in Table 1, but readers are referred to the GO web site and the documentation for the *GO.db* package for a more comprehensive listing. Clearly for some investigations one will want to exclude genes that were mapped according to some of the evidence codes.

IMP	inferred from mutant phenotype
IGI	inferred from genetic interaction
IPI	inferred from physical interaction
ISS	inferred from sequence similarity
IDA	inferred from direct assay
IEP	inferred from expression pattern
IEA	inferred from electronic annotation
TAS	traceable author statement
NAS	non-traceable author statement
ND	no biological data available
IC	inferred by curator

Table 1: GO Evidence Codes

In some sense TAS is probably the most reliable of the mappings. IEA is a weak association and is based on electronic information, no human curator has examined or confirmed this association. As we shall see later, IEA is also the most common evidence code.

The sets of mappings of interest are roughly divided into three parts. First there is the basic description of the terms etc., these are provided in the `GOTERMS` hash table. Each element of this hash table is named using its GO identifier (these are all of the form `GO:` followed by seven digits). Each element is an instance of the `GOTerms` class. A complete description of this class can be obtained from the appropriate manual page (use `class?GOTerms`). From these data we can find the text string describing the term, which ontology it is in as well as some other basic information.

There are also a set of hash tables that contain the information about parents and children. They are provided as hash tables (the `XX` in the names below should be substituted for one of `BP`, `MF`, or `CC`.

- `GOXXPARENTS`: the parents of the term
- `GOXXANCESTOR`: the parents, and all their parents and so on.

- GOXXCHILDREN: the children of the term
- GOXXOFFSPRING: the children, their children and so on out to the leaves of the GO graph.

For the GOXXPARENTS mappings (only) information about the nature of the relationship is included.

```
> GOTERM$"GO:0003700"
```

```
GOID: GO:0003700
Term: DNA binding transcription factor activity
Ontology: MF
Definition: Interacting selectively and non-covalently with a specific
    DNA sequence (sometimes referred to as a motif) within the
    regulatory region of a gene in order to modulate transcription.
Synonym: nucleic acid binding transcription factor activity
Synonym: sequence-specific DNA binding transcription factor activity
Synonym: transcription factor activity
Synonym: GO:0000130
Synonym: GO:0001071
Secondary: GO:0000130
Secondary: GO:0001071
```

```
> GOMFPARENTS$"GO:0003700"
```

```
        is_a
"GO:0140110"
```

```
> GOMFCHILDREN$"GO:0003700"
```

```
        is_a         is_a         is_a         is_a         is_a         is_a
"GO:0000981" "GO:0001010" "GO:0001011" "GO:0001034" "GO:0001130" "GO:0001167"
        is_a         is_a         is_a
"GO:0001199" "GO:0034246" "GO:0098531"
```

```
>
```

Here we see that the term GO:0003700 has two parents, that the relationships are is-a and that it has one child. One can then follow this chains of relationships or use the ANCESTOR and OFFSPRING hash tables to get more information.

The mappings of genes to GO terms is not contained in the GO package. Rather these mappings are held in each of the chip and organism specific data packages, such as hgu95av2GO and org.Hs.egGO are contained within packages hgu95av2.db and org.Hs.eg.db respectively. These mappings are from a Entrez Gene ID to the most specific applicable GO terms. Each such entry is a list of lists where the innermost list has these names:

- `GOID`: the GO identifier
- `Evidence`: the evidence code for the assignment
- `Ontology`: the ontology the GO identifier belongs to (one of `BP`, `MF`, or `CC`).

Some genes are mapped to a GO identifier based on two or more evidence codes. Currently these appear as separate entries. So you may want to remove duplicate entries if you are not interested in evidence codes. However, as more sophisticated use is made of these data it will be important to be able to separate out mappings according to specific evidence codes.

In this next example we consider the gene with Entrez Gene ID `4121`, this corresponds to Affymetrix ID `39613_at`.

```
> ll1 = hgu95av2GO[["39613_at"]]
> length(ll1)

[1] 17

> sapply(ll1, function(x) x$Ontology)

GO:0006486 GO:0006491 GO:1904381 GO:0000139 GO:0000139 GO:0005783 GO:0005783
      "BP"       "BP"       "BP"       "CC"       "CC"       "CC"       "CC"
GO:0005793 GO:0005794 GO:0005829 GO:0016020 GO:0016021 GO:0070062 GO:0004571
      "CC"       "CC"       "CC"       "CC"       "CC"       "CC"       "MF"
GO:0004571 GO:0005509 GO:0015923
      "MF"       "MF"       "MF"

>
```

We see that there are 17 different mappings. We can get only those mappings for the `BP` ontology by using `getOntology`. We can get the evidence codes using `getEvidence` and we can drop those codes we do not wish to use by using `dropECode`.

```
> getOntology(ll1, "BP")

[1] "GO:0006486" "GO:0006491" "GO:1904381"

> getEvidence(ll1)

GO:0006486 GO:0006491 GO:1904381 GO:0000139 GO:0000139 GO:0005783 GO:0005783
     "IEA"      "IBA"      "TAS"      "IBA"      "TAS"      "IBA"      "TAS"
GO:0005793 GO:0005794 GO:0005829 GO:0016020 GO:0016021 GO:0070062 GO:0004571
     "IDA"      "IDA"      "IDA"      "HDA"      "IEA"      "HDA"      "IBA"
GO:0004571 GO:0005509 GO:0015923
     "TAS"      "IEA"      "TAS"
```

```
> zz = dropECode(ll1)
> getEvidence(zz)

GO:0006491 GO:1904381 GO:0000139 GO:0000139 GO:0005783 GO:0005783 GO:0005793
     "IBA"      "TAS"      "IBA"      "TAS"      "IBA"      "TAS"      "IDA"
GO:0005794 GO:0005829 GO:0016020 GO:0070062 GO:0004571 GO:0004571 GO:0015923
     "IDA"      "IDA"      "HDA"      "HDA"      "IBA"      "TAS"      "TAS"

>
```

A Basic Description of GO

We now characterize GO and some of its properties. First we list some of the specific GO IDs that might be of interest (please feel free to propose even more).

- GO:0003673 is the GO root.
- GO:0003674 is the MF root.
- GO:0005575 is the CC root.
- GO:0008150 is the BP root.
- GO:0000004 is biological process unknown
- GO:0005554 is molecular function unknown
- GO:0008372 is cellular component unknown

We can find out how many terms are in each of the different ontologies by:

```
>  zz = Ontology(GOTERM)
>  table(unlist(zz))

    BP     CC     MF universal
 29595   4171  11150         1

>
```

Or we can ask about the number of is-a and partof relationships in each of the three different ontologies.

```
>  BPisa = eapply(GOBPPARENTS, function(x) names(x))
>  table(unlist(BPisa))
```

```
                is_a negatively_regulates                 part_of
               57238                 2798                    5392
positively_regulates            regulates
                2778                 3223
```

```
>  MFisa = eapply(GOMFPARENTS, function(x) names(x))
>  table(unlist(MFisa))
```

```
   is_a part_of
  13679      11
```

```
>  CCisa = eapply(GOCCPARENTS, function(x) names(x))
>  table(unlist(CCisa))
```

```
   is_a part_of
   6035    1458
```

```
>
```

Working with GO

Finding terms that have specific character strings in them is easily accomplished using `grep`. In the next example we first convert the data from `GOTERM` into a character vector to make it easier to do multiple searches.

```
>  goterms = unlist(Term(GOTERM))
>  whmf = grep("fertilization", goterms)
```

So we see that there are 15 terms with the string "fertilization" in them in the ontology. They can be accessed by subsetting the `goterms` object.

```
>  goterms[whmf]
```

```
                                                                                              "sing

                              "fusion of sperm to egg plasma membrane involved in sing



                                                   "double fertilization forming a zygo

                                                             "double fertilization for
```

```
                                        "fertilization, exchange of chro

                                                    "respiratory burst

                                                             "fertil

                                                 "negative regulation

          "fusion of sperm to egg plasma membrane involved in double fertilization for

"fusion of sperm to egg plasma membrane involved in double fertilization forming a zygo

                                                           "regulation

                             "regulation of double fertilization forming a zygo

           "male-female gamete recognition during double fertilization forming a zygo

                                                 "positive regulation

>
```

Working with chip specific meta-data

In some cases users will want to restrict their attention to the set of terms etc that map to genes that were assayed in the experiments that they are working with. To do this you should first get the appropriate chip specific meta-data file. Here we demonstrate some of the examples on the Affymetrix HGu95av2 chips and so use the package *hgu95av2.db*. Each of these packages has a data environment whose name is the basename of the package with a `GO` suffix, so in this case `hgu95av2GO`. Note that if there are many manufacturer ids that map to the same Entrez Gene identifier then these will be duplicate entries (with different keys).

We can get all the `MF` terms for our Affymetrix data.

```
> affyGO = eapply(hgu95av2GO, getOntology)
> table(sapply(affyGO, length))

   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15 
1680 1167 1554 1734 1596 1195  919  717  438  390  295  184  164  129  102   67 
  16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   32 
  56   42   37   31   21   11   27   10    5   13    7    6   10    4    4    4 
  35   37   39 
   1    2    3 
```

```
>
```

How many of these probes have multiple GO terms associated with them? What do we do if we want to compare two genes that have multiple GO terms associated with them?

What about evidence codes? To find these we apply a similar function to the affyGO terms.

```
> affyEv = eapply(hgu95av2GO, getEvidence)
> table(unlist(affyEv, use.names=FALSE))

  EXP   HDA   HEP   HMP   IBA    IC   IDA   IEA   IEP   IGI   IMP   IPI   ISA
  588  6514    80    98 38809  1307 57562 67701  1073  1506 18637 14028     2
  ISM   ISS   NAS    ND   TAS
  749 21405  7445   717 47671

>

> test1 = eapply(hgu95av2GO, dropECode, c("IEA", "NR"))
> table(unlist(sapply(test1, getEvidence),
+              use.names=FALSE))

  EXP   HDA   HEP   HMP   IBA    IC   IDA   IEP   IGI   IMP   IPI   ISA   ISM
  588  6514    80    98 38809  1307 57562  1073  1506 18637 14028     2   749
  ISS   NAS    ND   TAS
21405  7445   717 47671
```

These functions make is somewhat straightforward to select subsets of the GO terms that are specific to different evidence codes.

References

E. Camon, M. Magrane, D. Barrell, V. Lee, E. Dimmer, D. Binns J. Maslen, N. Harte, R. Lopez, and R. Apweiler. The Gene Ontology annotation (GOA) database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Research*, 32:D262–D266, 2004.

The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.