

Chapter 16

A Gene Ontology Tutorial in Python

Alex Warwick Vesztrocy and Christophe Dessimoz

Abstract

This chapter is a tutorial on using Gene Ontology resources in the Python programming language. This entails querying the Gene Ontology graph, retrieving Gene Ontology annotations, performing gene enrichment analyses, and computing basic semantic similarity between GO terms. An interactive version of the tutorial, including solutions, is available at <http://gohandbook.org>.

Key words Gene Ontology, Tutorial, Python

1 Introduction

One of the main goals of developing a formal ontology is to facilitate computational analysis. The purpose of this chapter is to provide a hands-on introduction to handling GO terms and GO annotations in Python. This tutorial also shows how Python can be used to perform GO term enrichment analyses, as well as how to compute the similarity between GO terms.

This tutorial uses Python, but other popular languages commonly used to perform GO analyses include Java, R, Perl, and Matlab. The Gene Ontology consortium website maintains a list of software libraries, accessible from

ftp://ftp.geneontology.org/pub/go/www/GO.tools_by_type.software.shtml

An interactive version of this tutorial, with model solutions to all the questions, is available from the book homepage at <http://gohandbook.org>.

2 Querying the Gene Ontology

A fundamental first step is to retrieve the Gene Ontology and analyse that structure (Chap. 3 [1]).

One convenient Python package available to query the GO is GOATOOLS [2]. This package can read the GO structure stored in OBO format, which is available from the GO website (see Chap. 11 [3]). After loading this file, it is possible to traverse the GO structure, search for particular GO terms, and find out which other terms they are related to and how.

This package is available on the Python Package Index (PyPI), a standard repository of python libraries. As such, it is possible to install it locally using the command¹:

```
pip install goatools
```

The GOATOOLS package contains the functions necessary to parse the GO in OBO format, to query it, and to visualise the ontology. Using the function `obo_parser.GODag()` from GOATOOLS, the GO file can be loaded. Each GO term in the resulting object is an instance of the `GOTerm` class, which contains many useful attributes, such as:

- `GOTerm.name`: textual definition;
- `GOTerm.namespace`: the ontology the term belongs to (i.e., Molecular Function [MF], Biological Process [BP], or Cellular Component [CC]);
- `GOTerm.parents`: list of parent terms;
- `GOTerm.children`: list of children terms;
- `GOTerm.level`: shortest distance to the root node;

Exercise 2.1

Download the GO basic file in OBO format (`go-basic.obo`), and load the GO using the function `obo_parser.GODag()` from GOATOOLS. Using this library, answer the following questions:

- (a) What is the name of the GO term `GO:0048527`?
- (b) What are the immediate parent(s) of the term `GO:0048527`?
- (c) What are the immediate children of the term `GO:0048527`?
- (d) Recursively find all the parent and child terms of the term `GO:0048527`. *Hint*: use your solutions to the previous two questions, with a recursive loop.
- (e) How many GO terms have the word “*growth*” in their name?
- (f) What is the deepest common ancestor term of `GO:0048527` and `GO:0097178`?
- (g) Which GO terms regulate `GO:0007124` (pseudohyphal growth)? *Hint*: load the relationship tags and look for terms which define regulation.

¹GOATOOLS version 0.6.4 was used to write this tutorial and the exercises. To install this exact version, use `pip install goatools==0.6.4`

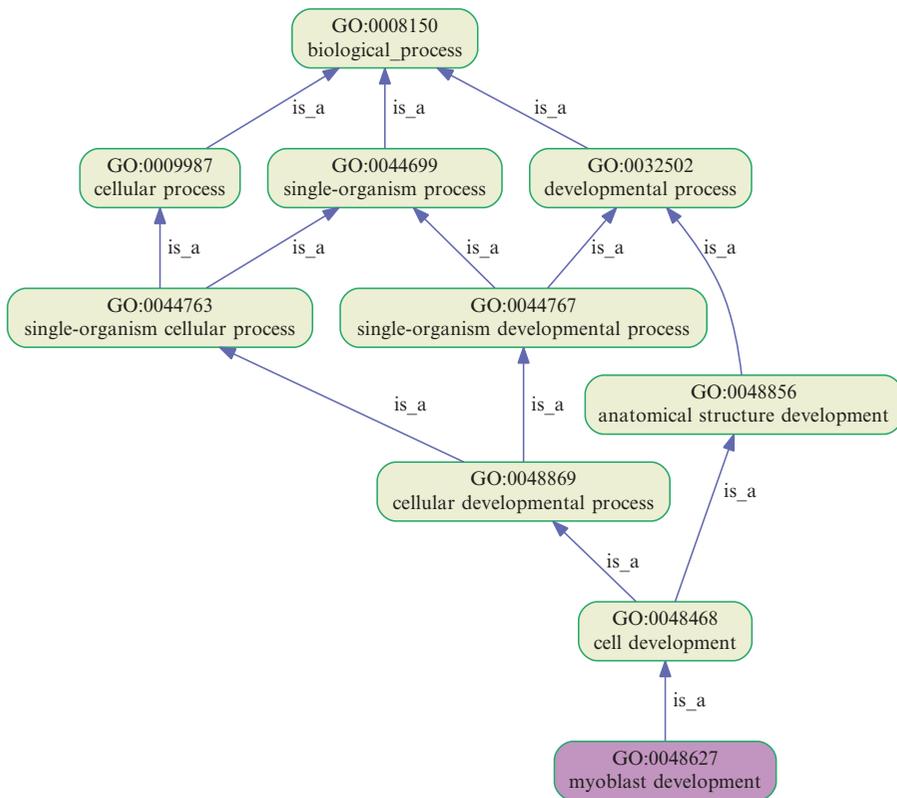


Fig. 1 Selected parts of the Gene Ontology can be visualised using the GOATOOLS library [2]

Exercise 2.2

Using the visualisation function in the GOATOOLS library, answer the following questions:

- Produce a figure similar to that in Fig. 1, for the GO term `GO:0097190`. From the visualisation, what is the name of this term?
- Using this figure, what is the most specific term that is in the parent terms of both `GO:0097191` (extrinsic apoptotic signalling pathway) and `GO:0038034` (signal transduction in absence of ligand)? This is also referred to as the lowest common ancestor (see Chap. 12 [4]).

Furthermore, other tag-value lines such as the “relationships” can be loaded with an optional argument of, e.g., `optional_attrs=['relationship']`.

The GOATOOLS library also includes functions to visualise the GO graph. For instance, it is possible to depict the location of a particular GO term in the ontology using the method `GOTerm.draw_lineage()`. For example, the plot in Fig. 1 showing the lineage of the GO term `GO:0048627` was created using this function.

As an alternative to GOATOOLS and OBO files, it is possible to retrieve information relating to a specific term from a web service. One such service is the EMBL-EBI QuickGO resource (see

Chap. 11; [3, 5]), which can provide descriptive information about GO terms in OBO-XML format. It is possible to request this OBO-XML file over HTTP, using a URL of the form

http://www.ebi.ac.uk/QuickGO/GTerm?id=<GO_ID>&format=oboxml

where <GO_ID> is replaced with the GO identifier for the term of interest. In Source Code 2.1, an example function to automate this in Python is listed, which uses the `urllib` library to request the OBO-XML and the `xmldict` library to parse the XML into an easy to use dictionary structure. Both libraries are available to install using `pip`, if required. Note that the `future` library was used to ensure that the function is both Python 2 and 3 compatible.

The dictionary structure that is returned can vary based on what information is available in the database. One example of an information-rich term is GO:0043065. A visualisation of the dictionary

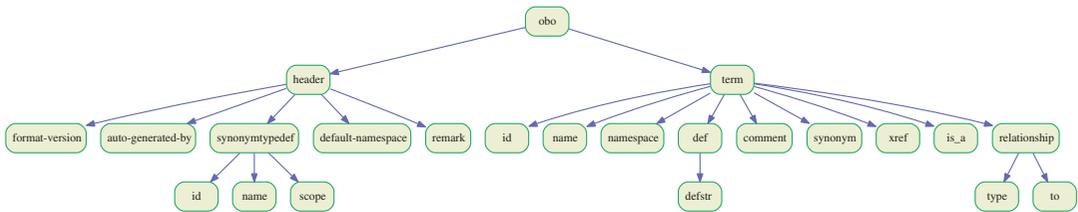


Fig. 2 Visualisation of the keys in the hierarchical dictionary structure returned by `get_oboxml('GO:0043065')`

Source Code 2.1. `get_oboxml()` function for Python 2 and 3.

```

from future.standard_library import install_aliases
install_aliases()
from urllib.request import urlopen
import xmldict

def get_oboxml(go_id):
    """
        This function retrieves the OBO-XML for a
        given Gene Ontology term, using EMBL-EBI's
        QuickGO browser.
        Input: go_id - a valid Gene Ontology ID,
        e.g. GO:0048527.
    """
    quickgo_url = "http://ebi.ac.uk/QuickGO/GTerm?id="+
    go_id+"&format=oboxml"
    oboxml = urlopen(quickgo_url)

    # Check the response
    if oboxml.getcode() == 200:
        obodict = xmldict.parse(oboxml.read())
        return obodict
    else:
        raise ValueError("Couldn't receive OBOXML
        from QuickGO. Check URL and try again.")
  
```

structure for this term, created with the `visualisedictionary` package available from PyPI (using `pip`), has been included in Fig. 2.

The main advantage of using a web service, such as QuickGO, is that there is no requirement to download and parse the entire Gene Ontology structure; only the information required is retrieved. This is therefore more efficient if only a few particular terms are involved in an analysis. By contrast, for analyses involving many terms, the file-based approach described above is more suitable.

Exercise 2.3

Using the function `get_oboxml()`, listed in Source Code 2.1, answer the following questions:

- (a) Find the name and description of the GO term `GO:0048527` (lateral root development). *Hint*: print out the dictionary returned by the function and study its structure, or use the visualisation in Fig. 2.
- (b) Look at the difference in the OBO-XML output for the GO terms `GO:00048527` (lateral root development) and `GO:0097178` (ruffle assembly), then generate a table of the synonymous relationships of the term `GO:0097178`.

3 Retrieving GO Annotations

This section looks at manipulating the Gene Association File (GAF) standard, using a parser from the BioPython package [6].

Firstly, a GAF file, which contains GO annotations, shall be downloaded from the UniProt-GOA database [7]. Their website (<https://www.ebi.ac.uk/GOA/downloads>) lists a number of variants. For this tutorial the reduced GAF file containing only the gene association data for *Arabidopsis thaliana* is going to be used.

Annotations from GAF files can be loaded into a Python dictionary using an iterator from the BioPython package (`Bio.UniProt.GOA.gafiterator`). Source Code 3.1 shows a simple example of this being used, in order to print out the protein ID for each annotation.

Source Code 3.1

```
from Bio.UniProt.GOA import gafiterator
import gzip

# filename = <LOCATION OF GAF FILE>
filename = 'gene_association.goa_arabidopsis.gz'

with gzip.open(filename, 'rt') as fp:
    for annotation in gafiterator(fp):
        # Output annotated protein ID
        print(annotation['DB_Object_ID'])
```

Recall that the latest GAF standard, version 2.1, has 17 tab-delimited fields, which are described in detail in Chap. 3 [1]. Some of them include:

- 'DB': the protein database;
- 'DB_Object_ID': protein ID;
- 'Qualifier': annotation qualifier (such as NOT);
- 'GO_ID': GO term;
- 'Evidence': evidence code.

Exercise 3.1

- (a) Find the total number of annotations for *Arabidopsis thaliana* with NOT qualifiers. What is this as a percentage of the total number of annotations for this species?
- (b) How many genes (of *Arabidopsis thaliana*) have the annotation GO:0048527 (lateral root development)?
- (c) Generate a list of annotated proteins which have the word “growth” in their name.
- (d) There are 21 evidence codes used in the Gene Ontology project. As discussed in Chap. 3 [1], many of these are inferred, either by curators or automatically. Find the counts of each evidence code in the *Arabidopsis thaliana* annotation file.

4 GO Enrichment or Depletion Analysis

As discussed in detail in Chap. 13 [8] one of the most common analyses performed on GO data is an enrichment (or depletion) analysis. In this tutorial, the `GOEnrichmentStudy()` function available in the GOATOOLS library (which has been seen in section 2) will be used.

The `GOEnrichmentStudy()` function requires the following arguments:

1. the background set of terms (also known as the “population set”), passed as a list of GO term IDs;
2. associations between proteins IDs and GO term IDs, passed as a dictionary with protein IDs as the keys and sets of associated GO terms as the values;
3. the Gene Ontology structure, i.e., the output by the `obo_parser()` function from GOATOOLS;
4. whether annotations should be propagated to all parent terms, (defined in terms of `is_a` tags, only), indicated by setting the optional boolean parameter `propagate_counts` to `True` (default) or `False`;

5. the significance level, indicated by setting the optional parameter `alpha` to the desired cut-off (default: 0.05);
6. the foreground set of terms (also known as “study set”), indicated by setting the parameter `study` to a list of GO term IDs;
7. the list of method(s) to be used to assess significance, indicated by setting the parameter `methods` to a list containing one or several of these elements:
 - (a) "bonferroni": Fisher’s exact test with Bonferroni correction for multiple testing;
 - (b) "sidak": Fisher’s exact test with Šidák correction for multiple testing;
 - (c) "holm": Fisher’s exact test with Holm–Bonferroni correction for multiple testing;
 - (d) "fdr": Fisher’s exact test, controlling the false discovery rate (see Chap. 13 [8]).

The function returns the list of over-represented and under-represented GO terms in the population set, compared to the background set.

Exercise 4.1

Perform an enrichment analysis using the list of genes with the “growth” keyword from exercise 3.1.c. Use the *Arabidopsis thaliana* annotation set as background, also from exercise 3.1, and the GO structure from exercise 2.1.

- (a) Which GO term is most significantly enriched or depleted? Does this make sense?
- (b) How many terms are enriched, when using the Bonferroni corrected p -value ≤ 0.01 ?
- (c) How many terms are enriched, when using the false discovery rate (a.k.a. q -value) ≤ 0.01 ?

5 Computing Basic Semantic Similarities Between GO Terms

In this section, the focus is on computing semantic similarity between GO terms, based on ideas presented in detail in Chap. 12 [4]. Semantic similarity measures enable us to quantify the functional similarity of genes annotated with GO terms.

Recall that semantic similarity measures are broadly separated in two categories: graph-based and information-theoretic measures. The former relies only on the structure of the Gene Ontology graph, whilst the latter also accounts for the information content of the terms.

One graph-based measure of semantic similarity, presented in Chap. 12 [4], is the inverse of the number of edges separating two

terms. It is possible to compute the minimum number of edges separating two terms (t_1, t_2) by first finding the deepest common ancestor (t_{DCA}). Then the difference in depth between each term and the deepest common ancestor can be used to calculate the minimum distance between the terms. i.e.,

$$\text{min_distance}(t_1, t_2) = \text{depth}(t_1) + \text{depth}(t_2) - 2 \times \text{depth}(t_{DCA})$$

Further, one example of an information-theoretic measure (see Chap. 12 [4]) is Resnik's similarity measure—the information content of the most informative common ancestor of the two terms in question. The information content of a term is defined as the negative logarithm of its probability, which can be estimated from the frequency of the term in the annotation database of choice.

Exercise 5.1

- (a) GO:0048364 (root development) and GO:0044707 (single-multicellular organism process) are two GO terms taken from Fig. 1. Calculate the semantic similarity between them based on the inverse of the semantic distance (number of branches separating them).
- (b) Calculate the information content (IC) of the GO term GO:0048364 (root development), based on the frequency of observation in *Arabidopsis thaliana*.
- (c) Calculate the Resnik similarity measure between the same two terms as in part a.

Acknowledgements

We thank Adrian Altenhoff, Debra Klopfenstein, and Haibao Tang for helpful feedback on the tutorial. CD acknowledges Swiss National Science Foundation grant I50654 and UK BBSRC grant BB/M015009/1. Open Access charges were funded by the University College London Library, the Swiss Institute of Bioinformatics, the Agassiz Foundation, and the Foundation for the University of Lausanne.

Open Access This chapter is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, duplication, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, a link is provided to the Creative Commons license and any changes made are indicated.

The images or other third party material in this chapter are included in the work's Creative Commons license, unless indicated otherwise in the credit line; if such material is not included in the work's Creative Commons license and the respective action is not permitted by statutory regulation, users will need to obtain permission from the license holder to duplicate, adapt or reproduce the material.

References

1. Gaudet P, Škunca N, Hu JC, Dessimoz C (2016) Primer on the gene ontology. In: Dessimoz C, Škunca N (eds) *The gene ontology handbook. Methods in molecular biology*, vol 1446. Humana Press. Chapter 3
2. Tang H, Klopfenstein D, Pedersen B et al (2015) GOATOOLS: tools for gene ontology, Zenodo
3. Munoz-Torres M, Carbon S (2016) Get GO! retrieving GO data using AmiGO, QuickGO, API, files, and tools. In: Dessimoz C, Škunca N (eds) *The gene ontology handbook. Methods in molecular biology*, vol 1446. Humana Press. Chapter 11
4. Pesquita C (2016) Semantic similarity in the gene ontology. In: Dessimoz C, Škunca N (eds) *The gene ontology handbook. Methods in molecular biology*, vol 1446. Humana Press. Chapter 12
5. Binns D, Dimmer E, Huntley R et al (2009) QuickGO: a web-based tool for Gene Ontology searching. *Bioinformatics* 25:3045–3046
6. Cock PJA, Antao T, Chang JT et al (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* 25:1422–1423
7. Huntley RP, Sawford T, Mutowo-Meullenet P et al (2015) The GOA database: gene Ontology annotation updates for 2015. *Nucleic Acids Res* 43:D1057–63
8. Bauer S (2016) Gene-category analysis. In: Dessimoz C, Škunca N (eds) *The gene ontology handbook. Methods in molecular biology*, vol 1446. Humana Press. Chapter 13