BIOINFORMATICS APPLICATIONS NOTE Vol. 23 no. 2 2007, pages 257–258
doi:10.1093/bioinformatics/btl567

Systems biology

Using GOstats to test gene lists for GO term association

S. Falcon* and R. Gentleman

Fred Hutchison Cancer Research Center, Program Computational Biology,1100 Fairview Avenue North, P. O. Box 19024, Seattle, WA 98109, USA

Received on October 16, 2006; accepted on November 3, 2006
Advance Access publication November 10, 2006
Associate Editor: Trey Ideker

ABSTRACT

Motivation: Functional analyses based on the association of Gene Ontology (GO) terms to genes in a selected gene list are useful bioinformatic tools and the GOstats package has been widely used to perform such computations. In this paper we report significant improvements and extensions such as support for conditional testing.

Results: We discuss the capabilities of GOstats, a Bioconductor package written in R, that allows users to test GO terms for over or under-representation using either a classical hypergeometric test or a conditional hypergeometric that uses the relationships among GO terms to decorrelate the results.

Availability: GOstats is available as an R package from the Bioconductor project: http://bioconductor.org

Contact: sfalcon@fhcrc.org

1 INTRODUCTION

Version 2.0 of the Bioconductor package GOstats has substantial improvements for testing the association between Gene Ontology (GO) terms, see GO Consortium (2000); and a given gene list. We have implemented a conditional hypergeometric test that uses the relationships among the GO terms, similar to that presented in Alexa *et al*. (2006), to address concerns that arise due to the hierarchical structure of GO. Many other substantial improvements have also been made that make the software easier to use and the results more informative.

In this paper we briefly describe the preprocessing steps required to construct inputs for the testing function, followed by a presentation of the algorithms used, and the structure of the return value. We demonstrate capabilities of the GOstats package using a microarray dataset Chiaretti *et al*. (2004) from a clinical trial in acute lymphoblastic leukemia (ALL). More details on the analysis of this dataset are available in the GOstats package vignette.

2 INPUTS

To perform an analysis using the hypergeometric-based tests, one needs to define a 'gene universe' (usually conceptualized as the number of balls in an urn) and a list of selected genes from that universe. While it is clear that the selected gene list determines the results of the analysis, the fact that the universe has a large effect on the conclusions is, perhaps, less obvious and correct specification of the universe is important.

For microarray data, one can use the unique gene identifiers assayed in the experiment as the gene universe. However, some arrays, such as those from Affymetrix, attempt to include probes for as much of the genome as possible and often contain multiple probes corresponding to a single gene. The multiple probe issue should be resolved so that each gene is represented only once. One might also want to consider reduction of the universe to exclude genes that are not expressed, if such a determination can be made, since arguments can be made against maintaining objects in the universe that cannot be selected.

The next step is to identify the subset of the universe that is considered interesting. In many applications, this set is constructed by finding differentially expressed genes. One might use a *t*-test, or an receiver operating characteristic (ROC) curve, or any of a large number of methods to identify such genes. Other methods for finding sets of interesting genes can also be used.

2.1 Non-specific filtering

To obtain the universe we often use the following procedure. First we estimate the variability across samples using the the inter-quartile range, or a similar statistic. We remove probes without sufficient variability across samples to be informative; probes with little variability across samples are inherently uninteresting as they provide no discriminatory power. We remove probes that are missing either Entrez Gene identifiers or do not map to any GO terms. Finally, we refine the universe to ensure that each Entrez Gene identifier maps to exactly one probe by selecting the probe with the largest IQR when two or more probes map to the same Entrez Gene ID.

There are many valid approaches to non-specific filtering that might be quite different from the procedure described above. However, it is important to avoid double counting genes. In our approach, a gene is represented by an Entrez Gene ID and so we must ensure that each Entrez Gene ID is represented by at most one probe.

2.2 Parameters

Often one wishes to perform many similar analyses using slightly different sets of parameters. The main interface to the Hypergeometric tests, `hyperGTest`, facilitates this pattern of use by taking a single parameter object as its argument. This parameter is an instance of class `GOHyperGParams`. Using a parameter class

*To whom correspondence should be addressed.

instead of individual arguments makes it easier to organize and execute a series of related analyses. For example, one can create a list of *GOHyperGParams* instances and perform the hypergeometric test on each using R's lapply function:

```
resultList ← lapply(lisOfParamObjs, hyperGTest)
```

Below, we create a parameter instance by specifying the gene list, the universe, the name of the annotation data package, and the GO ontology we wish to interrogate. For the example analysis, we have stored the vector of Entrez Gene identifiers making up the gene universe in entrezUniverse. The selected genes are stored in selectedEntrezIds. In addition, users can specify a *P*-value cutoff, whether the conditional hypergeometric calculation should be used, and whether the test should evaluate over or under-representation of GO terms.

```
> hgCutoff ←0.001
> params ← new(''GOHyperGParams'',
+     geneIds = selectedEntrezIds,
+     universeGeneIds = entrezUniverse,
+     annotation = ''hgu95av2'', ontology = ''BP'',
+     PvalueCutoff = hgCutoff, conditional = FALSE,
+     testDirection = ''over'')
```

3 GOstats CAPABILITIES

In the hypergeometric model, each term is treated as an independent classification. Each gene is classified according to whether or not it has been selected and whether or not it is annotated at a particular term. A hypergeometric probability is computed to assess whether the number of selected genes associated with the term is larger than expected.

The hyperGTest function provides an implementation of the commonly applied hypergeometric calculation for over or under-representation of GO terms in a specified gene list. This computation ignores the structure of the GO terms and treats each term as independent from all other terms.

Often an analysis for GO term associations results in the identification of directly related GO terms with considerable overlap of genes. This is because each GO term inherits all annotations from its more specific descendants. To alleviate this problem, we have implemented a method which conditions on all child terms that are themselves significant at a specified *P*-value cutoff. Our approach is similar to that propsed in GO consortium (2000). Given a subgraph of one of the three GO ontologies, we test the leaves of the graph, that is, those terms with no child terms. Before testing the terms whose children have already been tested, we remove all genes annotated at significant children from the parent's gene list. This continues until all terms have been tested.

4 OUTPUTS

The hyperGTest function returns an instance of class *GOHyperGResult*. Printing the result at the R prompt provides a brief summary of the test performed and the number of significant terms found.

```
> hgOver ← hyperGTest(params)
> conditional(params) ← TRUE
> hgCondOver ← hyperGTest(params)
> hgOver

Gene to GO BP test for over-representation
1217 GO BP ids tested (22 have P < 0.001)
Selected gene set size: 582
     Gene universe size: 2915
     Annotation package: hgu95av2
```

The *GOHyperGResult* instance returned by hyperGTest contains the *P*-value, odds ratio, expected gene count, and actual gene count for each term tested along with the vector of gene identifiers annotated at each term. It is also possible to retrieve a graph instance representing the GO DAG for further computation.

All result components can be accessed programatically using accessor functions (see the manual page for the *GOHyperGResult* class for details). Calling summary on the result produces a data.frame summarizing the results which can optionally be limited to a user-specified minimum *P*-value and/or minimum gene count for the terms. To make it easy for non-technical users to review the results, the htmlReport function generates an HTML file that can be viewed in any web browser. The output generated by htmlReport as called below is available at http://bioconductor.org/docs/papers/2006/GOstats

```
> htmlReport(hgCondOver, file = ''ALL_hgco.html'')
```

ACKNOWLEDGEMENTS

We thank Tony Chiang for his helpful bug reports and Wolfgang Huber for the suggestion of adding the odds ratio to the output table.

Conflict of Interest: none declared.

REFERENCES

Alexa,A. *et al.* (2006) Improved scoring of functional groups from gene expression data by decorrelating GO graph structure. *Bioinformatics*, **22**, 1600–1607.

Chiaretti,S. *et al.* (2004) Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, **103**, 2771–2778.

GO Consortium (2000), Gene Ontology: tool for the unification of biology. *Nat. Genetics*, **25**, 25–29.